Theoretical Aspects of Randomization in Computation

A Thesis
Presented to
The Academic Faculty

by

Nisheeth K. Vishnoi

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in
Algorithms, Combinatorics and Optimization

College of Computing
Georgia Institute of Technology
July 2004

Theoretical Aspects of Randomization in Computation

Approved by:

Professor Richard J. Lipton, Committee Chair

Professor Prasad Tetali

Professor Saugata Basu

Professor H. Venkateswaran

Professor Yan Z. Ding

Date Approved: June 8, 2004

To my parents

Smt. Kanak Vishnoi and Dr. Naresh K. Vishnoi

and

my lovely sisters:

Ira and Shalini

ACKNOWLEDGMENTS

I am indebted to Professor Richard Lipton for supervising this thesis. Under his guidance, I was able to complete all the work in this thesis in about two years. I feel fortunate to have him as my advisor. His original mind, his keen insight in approaching hard problems, and his vast knowledge (which extends well beyond computer science) have served as an inspiration for me throughout this thesis. I am particularly thankful to him for sharing with me his prowess in number-theoretic arguments. The effortless manner in which he can demonstrate some of the hard number-theoretic results, has inspired me to learn a great deal about this beautiful subject. Indeed, this knowledge has already started to pay its dividends, as is apparent in a substantial part of this thesis. I am also grateful to him for reinforcing his faith in my abilities, for constantly encouraging me to try difficult problems, and for being a great collaborator. I cannot thank him enough for all he has done for me in the three years that I have known him.

My fascination with randomness goes back to my undergraduate studies. Under the supervision of Professor Ketan Mulmuley and Professor Sundar Vishwanathan at IIT Mumbai, I studied randomized algorithms for linear programming. During this period, they introduced me to the elegant areas of randomized algorithms and the probabilistic method. This has had a tremendous impact on my view about computation and has laid the foundations for this thesis.

At Georgia Tech., I am fortunate to have been guided by Professor Milena Mihail, Professor Dhruv Mubayi and Professor Prasad Tetali for the first two years of my graduate studies. In those formative years, I have learnt a lot from them. I would especially like to thank Milena for being a great collaborator and for all her advice. Though the work I did with her is not a part of this thesis, I consider it equally important and interesting. Her eye for elegance has made a clear impression on my mind, and for that, I am grateful to her.

Words are not enough for me to express my gratitude for Professor Yan Ding and

Professor H. Venkateswaran. Their role in my graduate studies at Georgia Tech. has been immense. I have greatly benefitted from them through seamless discussions over the years. From technical discussions to general advice about life, their words have been extremely valuable to me. Technically, I owe a lot of what I learnt about cryptography and extractors to Yan, and that about complexity theory to Venkat. They have been my mentors, teachers, advisors as well as great friends, and I thank them for always being there.

I would also like to thank Professor Robin Thomas at Georgia Tech. He has been the best teacher I have ever had. His ability to convey an idea with appropriate mathematical rigor, and at the same time maintaining clarity of thought, is amazing. His influence on me reflects through my thought process, my oratory skills, and my technical writings.

A substantial part of this thesis has been co-authored with Nikhil Devanur, Subhash Khot and Dick Lipton. I am grateful to them for allowing me to use our results I obtained with them in this thesis.

During my five years as a graduate student, I feel lucky to have been supported by the administrative staff of the College of Computing at Georgia Tech. With great help from them, I could spend most of my time focussing on my research. In particular, I am thankful to Violeta and Talib for all the technical support, Lerverne for taking care of all my travel, and more recently, Mary Claire for her help with scheduling appointments and lunches with Dick, and taking care of my employment paperwork.

Finally, and most importantly, I would like to thank my support group. I cannot thank enough, my parents, Smt. Kanak Vishnoi and Dr. Naresh Vishnoi, and my two sisters, Shalini and Ira, for their constant support. They have always encouraged me to strive for excellence, and this has made the road easy for me. Often, they put my interests before theirs. I feel blessed to have such a family. I owe to them everything that I have achieved, and I will achieve in life. I dedicate this thesis to them and to all they have done for me.

I would also like to thank my friends and teachers over the years who have showed trust in me and excused me for my shortcomings. I thank them for providing the mental strength I needed to brave the low points in my life and sharing the joy. In particular, I thank Mohit Agnihotri, Prof. Gaur, Syed Qadri, Manu Sharma and Nikhil Devanur and Tejas Iyer.

TABLE OF CONTENTS

SUMMARY

Randomness has proved to be a powerful tool in all of computation. It is pervasive in areas such as networking, machine learning, computer graphics, optimization, computational number theory and is *necessary* for cryptography. Though randomized algorithms and protocols assume access to *truly* random bits, in practice, they rely on the output of *imperfect* sources of randomness such as pseudo-random number generators or physical sources. Hence, from a theoretical standpoint, it becomes important to view randomness as a resource and to study the following fundamental questions pertaining to it:

EXTRACTION: How do we generate *high quality* random bits from *imperfect* sources?

RANDOMIZATION: How do we use randomness to obtain efficient algorithms?

DERANDOMIZATION: How (and when) can we *remove* our dependence on random bits?

In this thesis, we consider important problems in these three prominent and diverse areas pertaining to randomness. In randomness extraction, we present extractors for *oblivious bit fixing sources*. In (a non-traditional use of) randomization, we have obtained results in machine learning (learning juntas) and proved hardness of lattice problems. In derandomization, we present a deterministic algorithm for a fundamental problem called *identity testing*. In this thesis, we also initiate a complexity theoretic study of Hilbert's 17th problem. Here identity testing is used in an interesting manner.

A common theme in this work has been the use of tools from areas such as number theory in a variety of ways, and often, the techniques themselves are quite interesting.

CHAPTER I

INTRODUCTION

1.1 Randomness in computation

The introduction of randomization in computation has dramatically changed the way we view computation in the last twenty years. Though randomization was being employed since the 1940's: in Monte Carlo simulations in nuclear physics, in hashing and in randomizing the data among other areas, it was not till the late 1970's that it was incorporated into computation. Since then, randomization has proved to be a powerful tool in all of computer science, computational number theory and optimization.

In computer science, it promptly pervaded areas such as networking, machine learning, computer graphics, databases and cryptography.[1] An attractive feature of randomness is that, for many important problems, it leads to solutions which are simpler and (provably) more efficient than their deterministic counterparts.

1.2 Randomness as a resource

Though randomized algorithms and protocols assume access to truly random bits, in practice, they use the output of *imperfect* sources of randomness such as pseudo-random number generators or physical sources. Hence, from a theoretical standpoint, it becomes important to view randomness as a resource and to study the following fundamental questions pertaining to it:

EXTRACTION: How do we generate *high quality* random bits from *imperfect* sources?

RANDOMIZATION: How do we use randomness to obtain efficient algorithms?[2]

DERANDOMIZATION: How (and when) can we *remove* our dependence on random bits?

[1] In fact, cryptography is not possible without randomness.

[2] This is a very general paradigm and we refer the reader to the excellent book by Motwani and Raghavan [89], which explores this in great detail.

1.3 Contribution of this thesis

In this thesis, we consider important problems in these three prominent and diverse areas in randomness. In randomness extraction, we present extractors for *oblivious bit fixing sources*. In (a non-traditional use of) randomization, we have obtained results in machine learning (learning juntas) and proved hardness of lattice problems. In derandomization, we present a deterministic algorithm for a fundamental problem called *identity testing*. In this thesis, we also initiate a complexity theoretic study of Hilbert's 17th problem. Here identity testing is used in an interesting manner.

A common theme in this work has been the use of tools from areas such as number theory in a variety of ways, and the techniques themselves are quite interesting. Next, a brief summary of the results is presented, along with a brief discussion of extraction and derandomiation.

1.3.1 Randomness Extractors

Roughly, randomness extractors are functions which (efficiently) convert *imperfect* sources of randomness to ones which are *almost* random. The goal is to be able to simulate algorithms assuming access to imperfect sources only, without substantial loss in the efficiency or the quality of the solution. In addition, these find applications in areas such as cryptography, where there is a need for high quality random bits.

The difficulty in extracting randomness is related to how one defines *imperfect* sources of randomness. The following two examples illustrate this important issue. The first is due to von Neumann [122]. Consider two independent random variables $\{X_1, X_2\}$ each of which is 1 with probability p and 0 with probability $1-p$. The value of p is unknown. In this case, extracting true random bits is easy. The following table presents the extractor function:

X_1X_2	$Y = \mathsf{Ext}(X_1, X_2)$	Probability
00	reject	$(1-p)^2$
01	0	$(1-p)p$
10	1	$p(1-p)$
11	reject	p^2

The von Neumann extractor

It follows that the random variable Y, conditioned on the event "not reject", is 0 and 1 with equal probability ($= p(1-p)$). Thus, in this case, with a slight loss in entropy, one can efficiently extract truly random bits.

On the other extreme, is the second example of an imperfect source, for which extracting even one bit is provably impossible. Consider the set $\{0,1\}^n$, of n bit strings. For a set $S \subseteq \{0,1\}^n$, let X_S be the random variable with support S, and which is uniformly distributed on S. Although the range of X_S is $\{0,1\}^n$, one cannot hope to extract more than $\log_2 |S|$ bits[3] from such a random variable. Consider the family of random variables $\{X_S\}_{|S| \leq 2^{n-1}}$. It is easy to see that for any function $f : \{0,1\}^n \mapsto \{0,1\}$, which is supposed to output a *nearly* random bit, there is a set S from the family $\{X_S\}_{|S| \leq 2^{n-1}}$ on which f is constant. Indeed, without loss of generality assume that $|f^{-1}(0)| \geq |f^{-1}(1)|$. Then $|f^{-1}(0)| \geq 2^{n-1}$. Let T be any subset of $|f^{-1}(0)|$ of cardinality 2^{n-1}. Then, $f(T) = 0$. Notice that the amount of randomness in T is at least $n-1$ still one cannot even extract one random bit. This establishes the fact that randomness extraction is not possible from such a family of random variables! One gets around this problem by allowing the extractor to have access to a small random *seed*.[4] In algorithmic situations, this is then taken care of by running the output of the extractor on all possible seeds. These imperfect sources turn out to be the most important for a variety of reasons and the reader is referred to the surveys by Nisan and Ta-shma [93] and Shaltiel [110] for more details.

[3] This is the so called *min-entropy* of X_S.

[4] If one has access to more than one independent sample from a random variable from this family, this impossibility argument no longer works, and it is possible to extract random bits deterministically, see [33, 12].

1.3.2 Oblivious bit fixing sources

For cryptographic purposes, the need for *deterministic* extraction still remains. Albeit, one needs to consider imperfect sources which are somewhat less general than the second example discussed above. Here, the most promising class of imperfect source of randomness is the so called *oblivious bit fixing source*, for which deterministic extraction seems possible:

An (n, k) oblivious bit fixing source is a imperfect random source that outputs n bits, out of which all but k random bits are fixed, and the remaining are chosen independently and randomly.

In Chapter 6, we give an (efficient) deterministic extraction scheme which outputs $\Omega(\log k)$ random bits from any (n, k) oblivious bit fixing source. The extractor is *very simple* and *efficient*. This matches a result due to Kamp and Zuckerman [55]. We believe that there is a lot to be done here and our work is of preliminary nature.

1.3.3 Randomization: Learning symmetric juntas

In machine learning, one is typically given access to an oracle which provides *random examples* of a concept, and the goal is to learn this unknown concept efficiently. In this setting, randomization is an integral part of the model.[5] An important problem in machine learning is to learn in the presence of irrelevant information. More formally, the problem is to learn a function f on n bits, which depends only on some unknown k bits. Such a function is called a *k-junta*. It is widely considered (see [16, 88]) that learning the class k-juntas is one of the most important open problem in the theory of uniform distribution learning. It has connections with learning DNF formulae and decision trees of super-constant size, see [25, 53, 83, 119, 120] for more details. The hardness of learning k-juntas has also been a basis of a cryptosystem in [17]. This gives another reason to investigate the complexity of this problem.

The first non-trivial result for learning juntas was obtained by Mossel, O'Donnell and Servedio in [88]. They gave an algorithm that takes time (roughly) $n^{0.704k}$ to learn the concept class k-juntas. They prove their result by looking at Fourier and $\mathbb{F}_2$ representations of boolean functions simultaneously, and proving an elegant dichotomy theorem concerning

[5] If one is allowed to choose the examples, typically, the problem becomes easier.

these representation.

Fourier based learning has enjoyed a tremendous success in the computational learning theory community. The primary reason for this is two-fold:

- Fourier coefficients of a boolean function contain all the information about it.
- Fourier coefficients are easy to compute in the standard (PAC) learning theory setting.

Roughly, if for a family of boolean functions $\mathcal{F}$, it can be shown that every $f \in \mathcal{F}$ has some Fourier coefficient of order at most t which is not zero, then one can learn this family in time n^t. Letting $t = k$, one obtains a trivial n^k time algorithm for learning k-juntas. It is conjectured in [88] that if a k-junta has zero Fourier coefficients for all sets of size up to $2k/3$, then there is way to fix some $2k/3$ bits of it, such that the resulting function, with these $2k/3$ bits fixed, is a parity function (or its complement) in the free bits. A proof of this conjecture would result in a learning algorithm for k-juntas that runs in time $n^{2k/3}$. There are examples for which this conjecture, if true, would be the best possible. For instance, let $f(x_1, \ldots, x_n) := \text{MAJORITY}(x_1, \ldots, x_{2k/3}) \bigoplus \left(x_{2k/3+1} \oplus \cdots \oplus x_k\right)$. However, there is a way to learn this function in time $n^{k/3}$. It has been suggested in [16], that for general juntas, $n^{k/3}$ may be close to the best one can hope for. Substantially new ideas seem to be needed to go beyond this $k/3$ barrier. Most likely, these will have consequences beyond learning juntas. In the dearth of such approaches, a natural question is:

> What are sub-families of k-juntas for which Fourier based techniques will lead to fast algorithms?

Another important thing to notice about this function is that there is a huge sub-group of the symmetric group (S_k) acting on it, namely $S_{2k/3} \times S_{k/3}$. In light of these considerations, the most natural, and arguably an important sub-family, that comes up is the class of symmetric k-juntas. This is the class of all k-juntas which are symmetric in the variables they depend on. Surprisingly, even for this family, the best known algorithm before this work ran in time $n^{2k/3}$ [88]. This is the motivation for us to study the following structural result about the Fourier representation of symmetric boolean functions:

What is the smallest t such that every symmetric boolean function on k variables has a Fourier coefficient of order at most t which is not zero ?

Notice that if f is the parity function, or its complement, then $t = k$. So this question is interesting only when one prohibits f to be one of these two functions. Let $\tau(k)$ be the smallest such t in this case. We study this extremal parameter and our main contribution is a proof of the following *self similar* nature of this question:

$$\text{If } \tau(l) \leq s, \text{ then for } k \geq k_0(l),\ \tau(k) \leq \tfrac{s+1}{l+1}k.$$

Coupled with preliminary computer based calculations for $\tau(l)$, for small values of l, we obtain that $\tau(k) \leq 3k/19$. We believe that these results are bound to improve substantially with more sophisticated calculations for $\tau(l)$. As of now, these imply a learning algorithm for the class of symmetric k-juntas in time about $n^{3k/19}$. This is already much better than the $n^{2k/3}$ bound by [88] and goes below the $n^{k/3}$ barrier suggested for the *almost* symmetric function $\text{MAJORITY}(x_1, \ldots, x_{2k/3}) \bigoplus \left(x_{2k/3+1} \oplus \cdots \oplus x_k\right)$.

Apart from improving the results for learning symmetric juntas, we consider our technical contribution in studying this extremal parameter τ of equal importance. Studying τ reduces to analyzing a system of Diophantine equations. As a first step, we simplify these Diophantine equations. This is done by moving to a representation which is equivalent to the Fourier representation, but seems much simpler for the application of number theoretic tools. Once this is done, we study these Diophantine equations over local fields and combine the solutions in a combinatorial manner. The following self-similarity of Pascal's Triangle plays an important role. Recall that the m-th row of Pascal's Triangle consists of binomial coefficients of the form $\binom{m}{j}$, for $0 \leq j \leq m$. If $m = lp$ for some l, and some prime p, then the values obtained by reducing this row modulo p, can be read off directly from the l-th row of Pascal's Triangle! We hope that our line of work will finally result in an efficient learning algorithm for symmetric juntas.

1.3.4 Randomized reductions: Hardness of lattice problems

Computational complexity theory has really benefitted since the introduction of randomness as a resource available to the standard (deterministic/non-deterministic) Turing machines. For instance, important notions such as Interactive Proofs and Probabilistically Checkable Proofs can be defined only in this model of computation. Even before these powerful ideas, randomization was used in proving computational hardness of problems, for instance, in the result of Valiant and Vazirani [117]. They proved that SAT remains hard even if the problem is to decide if the given SAT formula is unsatisfiable or has exactly one satisfying assignment. Albeit, the NP-hardness of this problem (called USAT) is proved under the assumption that NP$\not\subseteq$RP.

An important problem for which *only* randomized reductions are known, is to find the shortest vector in a lattice. Formally, an n-dimensional lattice $\mathcal{L}$ is a set of vectors $\{\sum_{i=1}^{n} a_i \mathbf{b}_i \mid a_i \in \mathbb{Z}\}$, where $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n \in \mathbb{R}^n$ is a set of independent vectors, called the *basis* for the lattice. The same lattice could have many bases. Given a basis for an n-dimensional lattice, the Shortest Vector Problem asks for the shortest, non-zero vector in the lattice. The length of the vectors can be measured in any ℓ_p norm ($p \geq 1$), and the corresponding optimization problem is denoted by SVP_p.

The Shortest Vector Problem has been studied since the time of Gauss ([43], 1801), who gave an algorithm for SVP_2 in 2-dimensions. In a celebrated result, Lenstra, Lenstra and Lovász [72] gave a polynomial time algorithm for approximating SVP_2 within factor $2^{n/2}$. This algorithm has numerous applications, e.g. factoring rational polynomials, breaking knapsack-based cryptosystems, checking the solvability by radicals and integer programming in a fixed number of variables. Since all ℓ_p norms are within factor $\sqrt{n}$ from the ℓ_2 norm, these algorithms give similar approximations for SVP_p, for any p. It is a major open problem whether SVP_2 has polynomial factor approximations that run in polynomial time.

Proving NP-hardness for any finite p (in particular $p = 2$) was an embarrassing open problem for a long time. A breakthrough result by Ajtai [5] in 1998 finally showed that SVP_2 is NP-hard. This result was strengthened by Micciancio [84], who showed that SVP_p is hard to approximate within some constant factor, specifically, $2^{1/p}$. Recently, Khot [62]

showed that for every $\delta > 0$, there is a constant $p(\delta)$ such that for $p \geq p(\delta)$, SVP_p is hard to approximate within factor $p^{1-\delta}$. Surprisingly, all these reductions are randomized and **no** deterministic hardness results are known.

In this thesis, we obtain new hardness results for hardness of approximation of SVP_p. This is an improvement over previous results for certain range of values of p. Roughly, we show that for $p \geq 46$, it is hard to approximate SVP_p (in polynomial time) within a factor of $4/3$, unless $\mathsf{NP} \subseteq \mathsf{ZPP}$. For $p \geq 112$, we obtain a hardness factor of $3/2$. Further, our proofs are very simple (compared to Ajtai and Micciancio's proofs), and like Khot's result, our result works under the assumption $\mathsf{NP} \not\subseteq \mathsf{ZPP}$. Ajtai and Micciancio's results require the assumption $\mathsf{NP} \not\subseteq \mathsf{RP}$.

The *hardness* of SVP is the basis of cryptosystems of Ajtai-Dwork [6]. This was furthered by Regev in [100], who used the hardness of approximating the Unique Shortest Vector Problem (denoted USVP). This is a variant of SVP, where in the exact version, we are guaranteed to have (upto sign) only one non zero shortest vector in the lattice. In the λ-approximate version, the only lattice vectors of length at most λ times the shortest vector are the ones parallel to it. The exact version of USVP_2 was proved to be NP-Hard (again under randomized reductions) by Kumar and Sivakumar [66], *a la* Valiant-Vazirani [117]. We extend their result for every ℓ_p norm. We show that for all $p \geq 1$, USVP_p is hard to compute exactly in polynomial time, unless $\mathsf{NP} \subseteq \mathsf{BPP}$.

1.3.5 Derandomization

In derandomization one studies when one can *remove* or *reduce* the dependence of computation on randomness. The most important question here is whether $\mathsf{P}=\mathsf{BPP}$? Roughly, this question asks how powerful do efficient algorithms[6] become in the presence of randomness? There is some evidence that $\mathsf{P}=\mathsf{BPP}$. A result of Impagliazzo and Wigderson [52] states that if *sufficiently hard* to compute boolean functions exist, then $\mathsf{P}=\mathsf{BPP}$. In the other direction it is known that derandomizing BPP (to even quasi-polynomial time) implies strong circuit lower bounds for NEXP. Hence, establishing the close connection between randomness and

[6]Here efficiency is measured in terms of time. There is also a lot of interest in derandomizing space bounded algorithms [106].

hardness.

Among other results, it is known [71] that if P=NP, then P=BPP. Hence, proving P≠BPP implies P≠NP. The Karp-Lipton Theorem [58] suggests that it is unlikely that NP⊆BPP. Hence, these results point to the possibility that randomization is not as powerful as non-determinism. One of the immediate goals in derandomization is to establish (unconditionally) that BPP≠NEXP.

A very fruitful approach in derandomization has been to study specific randomized algorithms and try to remove randomness from them. This has resulted in the construction of sophisticated and general purpose tools, such as, small bias probability spaces, expanders, error correcting codes, dispersers, extractors and pseudo-random generators, to name a few. These have had implications far beyond their originally intended applications. An important instance of such a problem is the so called *polynomial identity testing* problem which we discuss next.

1.3.6 Polynomial identity testing

The polynomial identity testing problem is the following:

Given access *to two polynomials f and g, check whether f is identically equal to g.*

The complexity of this problem depends on what representation of f and g are given to us, as well as, the domain over which the polynomials are defined.[7] There is an efficient randomized algorithm for this problem. For more than 25 years, the existence of an efficient deterministic solution to this problem has remained open. Many fundamental problems, such as matching, program checking and primality testing can be reduced to it. Most of these applications, which reduce to testing whether a polynomial is identically zero over a field, rely only on a *black box* access to the polynomial. A derandomization for primality was recently discovered by Agarwal, Kayal and Saxena in their famous paper "PRIMES is in P" [3]. They showed that testing whether an integer n is a prime reduces to testing the identities of the kind $(a + x)^n = a + x^n \pmod{n}$, for which they gave a deterministic

[7] Notice that this problem is equivalent to the one where one is given just one polynomial f and it has to be decided whether f is identically zero.

polynomial time algorithm.

More recently, Kabanets and Impagliazzo [54] proved that derandomizing identity testing, in the arithmetic circuit setting, implies proving circuit lower bounds for NEXP. Since proving circuit lower bounds is a notoriously hard problem, it becomes important to investigate natural families of polynomials for which identity testing can be done in polynomial time.

In this thesis we present a simple and efficient deterministic algorithm for testing a fairly general class of polynomials for identity. More specifically, our algorithm can test if a given polynomial is identically zero in time which is a polynomial in: its *sparsity* and the *logarithm of its degree*. We just rely on a black box access to the polynomial and also prove near optimal lower bounds in this setting. Technically, our result is number theoretic in nature and uses deep facts about distribution of the smallest primes in arithmetic progressions. Our results are an improvement on the result of Klivans and Spielman in [65]. The difference is that in their case the dependence of the running time of the algorithm is on the degree of the polynomial, compared to the logarithm of the degree in our case.[8]

1.3.7 The complexity of Hilbert's 17th problem

As an important application of identity testing, we obtain a connection with Hilbert's 17th problem. Hilbert asked the following problem in his famous 1900 lecture:

Given a multi-variate polynomial that takes only non-negative values over the reals, can it be represented as a sum of squares of rational functions?

In 1927, E. Artin gave an affirmative answer to this question. His result guaranteed the existence of such a finite representation and raised the following important question:

What is the **minimum number** *of rational functions needed to represent any non-negative n-variate, degree d polynomial?*

In 1967, Pfister proved that any n-variate non-negative polynomial over the reals can be written as sum of squares of at most 2^n rational functions. In spite of a considerable effort by mathematicians for over 75 years, it is *not* known whether $n + 2$ rational functions are

[8] Preliminary version of our results appeared in [79].

sufficient! In lieu of the lack of progress towards the resolution of this question, we initiate the study of Hilbert's 17th problem from the point of view of computational complexity. In this setting, the following question is a natural relaxation:

What is the **descriptive complexity** *of the sum of squares representation (as rational functions) of a non-negative, n-variate, degree d polynomial?*

We consider arithmetic circuits as a natural representation of rational functions. We are able to show, assuming a standard conjecture in complexity theory, that it is impossible that every non-negative, n-variate, degree four polynomial, can be represented as a sum of squares of a *small* (polynomial in n) number of rational functions, each of which has a *small* size arithmetic circuit (over the rationals) computing it.

Our result points to the direction that it is unlikely that every non-negative, n-variate polynomial over the reals can be written as a sum of squares of a polynomial (in n) number of rational functions. Further, relating to standard (and believed to be hard to prove) complexity-theoretic conjectures sheds some light on why it has been difficult for mathematicians to close the $n + 2$ and 2^n gap. We hope that our line of work will play an important role in the resolution of this question.

1.4 Structure of the thesis

We have chosen to organize our results in the thesis chronologically. Chapter 2 presents our results about polynomial identity testing. This is based on a paper with Dick Lipton [79]. In Chapter 3 we present the connection to Hilbert's 17th problem. This is based on a manuscript[9] with Nikhil Devanur and Dick Lipton [38]. In Chapter 4 we present our results on the junta problem. This is based on an earlier unpublished article [37] and the manuscript [121]. Chapter 5 concerns the complexity of lattice problems and is based on a manuscript with Subhash Khot [64]. Finally, in Chapter 6 we present our results on oblivious bit fixing extractors. This is based on a preliminary draft with Dick Lipton [78].

[9] All the manuscripts have been submitted for publication at the time of the submission of this thesis.

CHAPTER II

POLYNOMIAL IDENTITY TESTING

2.1 Introduction

The problem of testing whether a multivariate polynomial $f(x_1, \ldots, x_n)$ is identically zero is extremely importance in theoretical computer science. Applications include *designing fast randomized algorithms*: most notably for program checking [20, 77] and perfect matching in graphs [30, 80, 90], *complexity theory*: in proving results such as IP=PSPACE [10, 82, 111] and PCP=NP [8]. Applications to lower bounds include a result of Kabanets and Impagliazzo [54], who show that derandomizing identity testing implies circuit lower bounds.

Recently, results on identity testing were used in proving PRIMES$\in$P by Agrawal, Kayal and Saxena [3]. They showed that testing whether an integer n is a prime reduces to efficiently testing the identities of the kind $(a + x)^n = a + x^n \pmod{n}$, for which they gave a deterministic polynomial time algorithm.

Another interesting application of polynomial identity testing was given by Devanur, Lipton and Vishnoi [38]. They relate the complexity of Hilbert's 17th problem to standard complexity classes[1]. Further applications and discussions about identity testing can be found in the text of Motwani and Raghavan [89].

To make the problem more precise, one needs to fix a representation of the given polynomial. If the polynomial is given as a list of its coefficients, the problem is trivial. Often, we are given an implicit representation of the polynomial, such as the determinant of a matrix or as an arithmetic circuit. It is easy to see that there *exists* a set S, with size polynomial in s and d, such that every non-zero multivariate polynomial of total degree at-most d, which can be described using s bits, evaluates to a non-zero value at at-least one of the points of S. Finding such a set deterministically seems elusive, and results of Kabanets and Impagliazzo

[1] Details appear in Chapter 3.

[54] seem to substantiate this difficulty, even for determinants.

2.1.1 Basic definitions and notations

Before continuing further, for the ease of presentation, we introduce some basic definitions. A polynomial is written as $f = \sum_\alpha c_\alpha \mathbf{x}^\alpha$, where $\mathbf{x}^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$. [2] A monomial is said to appear in f, if the corresponding coefficient is non-zero. $m(f)$ denotes the number of monomials in f. $d_i(f)$, for $1 \leq i \leq n$, is the degree of x_i in f. $d(f) := \max_i d_i(f)$ is called the **maximum degree** of f. $\delta(f)$ denotes the maximum **total degree** of f. $H(f)$ denotes the **height** of f, that is, if $f = \sum_\alpha c_\alpha \mathbf{x}^\alpha$, then $H := \max_\alpha |c_\alpha|$. By abuse of notation, we will use m, d_i, d, δ, H to denote these quantities for f when the context is clear. All logarithms are base 2, unless stated otherwise.

A polynomial $f = \sum_\alpha c_\alpha \mathbf{x}^\alpha$ (defined over a ring of characteristic zero) is said to be identically zero, denoted $f \equiv 0$, if $c_\alpha = 0$, for all α.

A *Black Box* computing a polynomial $f(x_1, \ldots, x_n) \in \mathbb{Z}[x_1, \ldots, x_n]$ is an oracle that evaluates f at a given point $(a_1, \ldots, a_n) \in \mathbb{Z}^n$. A *Promise Black Box* is a black box with a given set of promises Π_0 and Π_1, computing a polynomial f that satisfies one of the promises Π_0 or Π_1. Typically Π_0 will be the predicate that $f \equiv 0$ over $\mathbb{Z}^n$, while Π_1 a set of conditions on the polynomial: e.g. bounding its height, number of monomials, degree etc. Typical notation will look like $\mathsf{PBB}^f[0, (n, m, d, H)]$, which means that either $f \equiv 0$ or f is a polynomial in n variables with at most m monomials, maximum degree d and height H.

2.1.2 Previous work

The main ideas in polynomial testing evolved over a period of 25 years, but most notable progress was made in the last decade. In this section we sketch a brief history of important contributions.

DeMillo-Lipton '78, Zippel 79, Schwartz '80

The earliest work on zero testing can be traced to DeMillo and Lipton [36], Schwartz [109] and Zippel [124] back to late 1970's. They use the fact that for a non-zero polynomial

[2] In such a summation no monomial is repeated more than once. We will just be concerned with polynomials defined over rings of zero characteristic.

$f(x_1, \ldots, x_n) \in R^n$, if we choose a big enough set from R and set the variables randomly and independently from it, with high probability, f does not evaluate to zero. Formally

Theorem 2.1.1. *For $f \not\equiv 0 \in R[x_1, \ldots, x_n]$, for any set $S \subseteq R$, and a_i chosen randomly and independently from S,*

$$\Pr[f(a_1, \ldots, a_n) = 0] \leq \frac{\delta(f)}{|S|}.$$

If we allow $\delta(f)$ to be a part of the input, this theorem establishes that identity testing is in co-RP. For circuits, the degree of the polynomial computed by a circuit of size s could be as big as 2^s. With a simple modification, Ibarra and Moran [51] established a similar result when f is given as an arithmetic circuit over the integers.

Chen-Kao '97, Lewin-Vadhan '98

In a remarkable paper, Chen and Kao [31] show how to reduce randomness using irrational numbers. They use the fact that if one substitutes algebraic numbers of sufficiently high degree (over $\mathbb{Q}$) in a polynomial with integer coefficients, a non-zero polynomial will not evaluate to zero. Of course, one cannot substitute algebraic numbers, but only their rational approximations. They use random bits to chose a random *conjugate* of the algebraic numbers at hand and plug in its suitable rational approximation. They need $\sum_{i=1}^{n} \lceil \log(d_i + 1) \rceil$ random bits, where d_i is the degree of the polynomial in x_i. Contrary to the classic algorithms, the error probability is reduced not by increasing the number of random bits, but by increasing the precision of the rational approximations. Thus, letting the algorithm run for more time reduces error. Further, their proof assumes just a black box access to f.

This technique was generalized by Lewin and Vadhan [75] to work over any field, when f is given as a *straight line program*. If f is given as a black box, they needed the field to be large enough. They also showed the optimality of the random bits used in the black box model.

Agrawal-Biswas '99, Agrawal-Kayal-Saxena '02

Further improvements were made in the work of Agrawal and Biswas [2]. They showed how to do identity testing with $\lceil \sum_{i=1}^{n} \log(d_i+1) \rceil$ random bits. (Notice that the ceiling is outside the summation, compare to Chen-Kao.) But they assumed a circuit representation of f and their algorithm works over the ring $\mathbb{Z}_t$ or $\mathbb{Q}$. But more importantly, they proposed a test for primality based on the identity mentioned in the introduction. Finally a breakthrough occurred in 2002, when Agrawal, Kayal and Saxena [3] gave a deterministic algorithm for testing this identity and hence establishing that $\mathsf{PRIMES} \in \mathsf{P}$.

Klivans-Spileman '00

Klivans and Spielman [65] used ideas from error correcting codes and the Isolation Lemma [30, 90] to give a randomized polynomial time algorithm which uses $O(\log mn\delta)$ random bits for $\mathsf{PBB}^f[0, (n, m, \delta, \cdot)]$. Up to constants, their result generalizes the previously mentioned results of [2, 31, 79]. This algorithms implies a deterministic algorithm to test whether a *sparse* polynomial is identically zero or not. The time taken by this deterministic algorithm is polynomial in m, n and δ. Hence, if m and δ are polynomial, this is a polynomial time algorithm. This also has an application in learning theory [65], where one needs to interpolate sparse polynomials.

Kabanets-Impagliazzo '03

A result by Kabanets and Impagliazzo [54] asserts that identity testing of polynomials given as circuits over integers is *essentially* as hard as proving circuit lower bounds for NEXP. They also get a partial converse to this statement: one can deterministically test a circuit (over integers) of size poly(n), computing a polynomial in n variables, of degree at most poly(n), in sub-exponential time, if $\mathsf{PERMANENT}$ has super polynomial arithmetic circuit complexity. The latter part is an analogue of the classic hardness-randomness tradeoff for boolean functions introduced by Nisan and Wigderson [94]. Their results, along with a result of Valiant [54], also implies hardness of derandomizing identity testing for determinants.

2.1.3 Our results and techniques

Algorithmic results

The main result of this chapter is a deterministic algorithm for testing identities over integers for $\mathsf{PBB}^f[0,(n,m,d,H)]$. The algorithm runs in time $\text{poly}(mn\log(d+1)+\log H)$. This improves on Klivans and Spielman: the dependence of the running time here is *logarithmic* in the *maximum degree*, while it is *polynomial* in *total degree* in case of Klivans and Spielman. Another important feature of our algorithm is that it is *very simple.* However, the proof of correctness relies on deep results from analytic number theory about the distribution of primes in arithmetic progressions.[3]

This chapter also contains lower bounds relating to polynomial identity testing. We outline them below.

Lower bounds

In light of the result of Kabanets and Impagliazzo [54], lower bounds for identity testing seem plausible only for certain special models, such as the Black Box model. This model is ubiquitous, most notably in cryptographic applications, [22], hence it seems important to understand its limits. By a lower bound here one means the number of queries any deterministic algorithm can be forced to make, before it is certain that the box represents a zero polynomial. This bound translates easily into a randomized lower bound: the minimum number of random bits an algorithm needs to decide the same. The minimum number of random bits needed is roughly the logarithm of the minimum number of queries any deterministic algorithm needs to make, see [75]. (Henceforth we only talk about deterministic lower bounds.)

Most positive results about identity testing use the given polynomial as a black box. Lewin and Vadhan [75] gave a simple lower bound argument which matched (up to constants) their algorithm. Their proof is essentially a dimensionality argument on the vector space of monomials. When a promise bound on the height of the polynomial is given, the

[3]Results using similar techniques were reported by Shparlinski [112] in some of his unpublished work with Karpinski.

dimension argument fails to give any meaningful answer for the query complexity.

In Section 2.3.2 we supply near matching lower bounds to the positive results in this chapter. (See Theorem 2.3.1.) This assumes a promise on the black box's height and the number of monomials. In the same section, we also supply optimal lower bounds for a technique developed by Chen and Kao [31]. (See Theorem 2.3.2.) The difference is that their algorithm checks *approximate equality* rather than equality: evaluate f and test if it is *close* to zero, rather than equal to zero.

2.2 Deterministic identity testing algorithm

The main result of this chapter is the following:

> Given a black box access to a multivariate polynomial $f(x_1, \ldots, x_n) \in \mathbb{Z}[x_1, \ldots, x_n]$, there is a deterministic algorithm which decides whether f is identically zero, and runs in time polynomial in $m, n, \log(d+1)$ and $\log H$. Moreover the bit lengths of the queries to the black box are logarithmic in $m, n, \log(d+1)$ and $\log H$.

The input bit lengths are important, as if we do restrict them, there is an easy algorithm to test identity. This is illustrated in a later section on the number of roots of sparse polynomials. Although we use some deep results from analytic number theory, the main idea is a clever use of the following mantra:

> A positive integer s can have at most $\log s$ distinct prime factors.

As mentioned before, via a different approach, Klivans and Spielman [65] obtain similar results. But their result implies a deterministic identity test which runs in time polynomial in m, n, d compared to ours, which runs in time polynomial in $m, n, \log H$ and $\log(d+1)$. So for exponential degree sparse polynomials, Klivans-Spielman implies an exponential algorithm, while ours is still a polynomial time algorithm. The main feature of our algorithm is that it is conceptually simpler than the one by Klivans and Spielman.

2.2.1 Number theory and identity testing

At the heart of our algorithm lies a number theoretic result, which we review next. It is well known, that for integers $(a, q) = 1$, any arithmetic progressions of the type $\{kq + a | k \geq 0\}$ contains infinitely many primes. This is a celebrated result of Dirichlet [103]. Analogous to the Prime Number Theorem (PNT), it is also known that the density of primes in such a progression is about $\frac{1}{\phi(q)}$. From the computational point of view, it is important to know the size (in terms of q) of the smallest prime in this progression. Let us consider a special arithmetic progression $\{kq + 1 | k \geq 0\}$, where q is a prime. Denote the smallest prime in this arithmetic progression by $p(q)$. Linnik [76] obtained an *unconditional*[4] result showing that there is an absolute constant L (called Linnik's constant), such that $p(q) < q^L$.[5] After decades of research, the best known value for L is 5.5. Even assuming the Generalized Riemann Hypothesis, the best one can hope for is $L = 2$.

Another result in this direction was recently proved by Baker and Harman [11]. They proved that on an average the Linnik's constant is below 2! In fact about 1.924. It turns out that this result is sufficient for our purpose! Further, this gets us a significant saving in running time while maintaining the simplicity of the algorithm of [79]. But for simplicity, we will also present the one using Linnik's Theorem. Technically, we obtain the following as a corollary of the Baker-Harman and Linnik's Theorem.

Theorem 2.2.1. *There is a fixed t_0 such that for all $t \geq t_0$, the set of primes less than $t^{1.924}$ contains at-least t distinct primes $p_1 < \cdots < p_t$, such that each $p_i = p(q_i)$ for some prime q_i. (Each $q_i < t^{1.924}$.) (Here for a prime q, $p(q)$ is the smallest prime in the arithmetic progression $\{kq + 1 | k \geq 0\}$.)*

Using this fact we prove the following theorem, which is an improvement on the result in [79]. We hope that in future more applications of this result will be discovered.

Theorem 2.2.2. *Given a black box access to a multivariate polynomial $f(x_1, \ldots, x_n) \in \mathbb{Z}[x_1, \ldots, x_n]$, there is a deterministic algorithm which decides whether f is identically zero*

[4] Without resorting to any standard conjectures in number theory such as the Riemann Hypothesis.
[5] He proved his result for more general arithmetic progressions.

in time polynomial in $O((mn\log(d+1)+\log H)^{3.848})$. *Moreover the bit lengths of the queries to the black box are logarithmic in* $m, n, \log(d+1)$ *and* $\log H$.

Organization of this section

In Section 2.2.2 we state some number theoretic preliminaries. In Section 2.2.3 we present our results related to primes in arithmetic progressions. In Section 2.2.4 we present the algorithm followed by its analysis in Section 2.2.5.

2.2.2 Number theoretic preliminaries

For integers a, d, (a, d) denotes their greatest common divisor. For $d > 0$, let $\phi(d)$ be the Euler's totient function of d, or the cardinality of the set $\{a|(a, d) = 1, 1 \leq a \leq d\}$. For a prime p, $\phi(p) = p - 1$.

The following simple lemma lies at the heart of our algorithm.

Lemma 2.2.3. *The number of distinct prime divisors of an integer* $s \geq 1$ *is at most* $\log_2 s$.

Proof. Let $p_1, \ldots, p_t$ be distinct prime divisors of s. By definition, for each i, $p_i \geq 2$. Since each p_i divides s, $s \geq \prod_{i=1}^{t} p_i \geq 2^t$. Hence $t \leq \log_2 s$. □

For a prime p let GF_p be the finite field of size p. Also, let $\mathsf{GF}_p^\times$ be the multiplicative sub-group of non-zero elements of GF_p. The following is a direct corollary of Fermat's Little Theorem and will be used later.

Lemma 2.2.4. *For a prime* p, *and integers* $s, t \geq 0$, *the polynomials* x^s *and* x^t *are the identical over* GF_p *if and only if* $p - 1$ *divides* $(s - t)$.

Proof. By Fermat's Little Theorem, in GF_p, the roots of the polynomial $x^{p-1} - 1$ are exactly the non-zero elements of GF_p. Write $x^s - x^t = x^t(x^{s-t} - 1)$. If $s - t = u(p - 1)$ for some integer u, then $x^{s-t} - 1 = x^{(p-1)u} - 1 = (x^{p-1} - 1)(1 + x^{p-1} + \cdots + x^{(p-1)(u-1)})$. Hence by the observation above, $x^{s-t} - 1$ is identically zero over $\mathsf{GF}_p^\times$, which in turn implies that x^s is identical to x^t over GF_p.

For the other direction, if $x^s - x^t$ is identically zero over GF_p, $x^{s-t} - 1$ must be identically zero over $\mathsf{GF}_p^\times$. Hence $x^{s-t} - 1$ must be divisible by $x^{p-1} - 1$. It follows that $p - 1$ should divide $s - t$. □

2.2.3 Distribution of primes in arithmetic progressions

The following lemma follows easily from the Prime Number Theorem or even relaxed versions of it.

Lemma 2.2.5. *[103] There is a constant $c > 0$ such that the n-th prime $p_n \leq cn \log n$.*

In this chapter, we are interested in the distribution of primes in arithmetic progressions. The most natural question here is:

Are there infinitely many primes in any arithmetic progressions?

Of course, one needs that the gcd of the difference and the starting term in the arithmetic progression be relatively prime. The following theorem establishes that apart from this trivial case, all arithmetic progressions contain infinitely many primes.

Theorem 2.2.6 (Dirichlet's Theorem [103]). *For integers $d \geq 2$ and $a \geq 1$, with $(a, d) = 1$, there are infinitely many primes in the arithmetic progression $\{a + kd | k \geq 0\}$.*

The next natural question is to investigate if there is an analogue of Lemma 2.2.5 for the case of arithmetic progressions. Before we address this question, we need some notation. For integers $d \geq 2$ and $a \geq 1$, with $(a, d) = 1$, let $p(d, a)$ be the smallest prime in the arithmetic progression $\{a + kd | k \geq 0\}$. Define

$$p(d) = \max_{1 \leq a < d, (a,d)=1} p(d, a).$$

Under the widely believed Generalized Riemann Hypothesis (GRH), Heath-Brown [103] proved that there is a constant c such that

$$p(d) \leq c(\phi(d))^2 (\log d)^2.$$

Notice that even if one believes the GRH, the best upper bound one can hope for is about d^2. In [76], Linnik managed to prove a weaker version of this fact, albeit, *without* any assumptions.

Theorem 2.2.7 (Linnik's Theorem [76]). *There is a constant $L > 1$ (called Linnik's constant) such that for every sufficiently large $d \geq q_0$,*

$$p(d) < d^L.$$

The best known value for L is 5.5 [103], while Schinzel, Sierpinski, and Kanold [103] have conjectured the value value to be 2. We mention here that it follows from the Prime Number Theorem that for every $\epsilon > 0$ and large enough d, $p(d) > (1-\epsilon)\phi(d)\log d$. For a detailed discussion on these facts the reader is referred to the book by Ribenboim [103].

It turns out that if we investigate Linnik's constant *on an average*, we can beat the bound suggested by GRH. First we need some more notation. Let $\pi(x)$ denotes the number of primes less than or equal to x and $\pi(x;q,a) := \sum_{1 \leq p \leq x,\ p \equiv a (\mathrm{mod}\ q)} 1$, where the summation is over primes p. The following (**unconditional**) theorem proved by Baker and Harman [11] captures the notion of Linnik's constant on an average[6].

Theorem 2.2.8 (Baker-Harman Theorem [11]). *There are functions $C_1(\theta), C_2(\theta)$, $(0.5 \leq \theta \leq 0.6)$ such that, for* **most** *$q \leq x^\theta$, (with $(a,q) = 1$)*

$$\frac{C_1(\theta)x}{\phi(q)\log x} < \pi(x;q,a) < \frac{C_2(\theta)x}{\phi(q)\log x}.$$

These functions satisfy

1. *$C_2(\theta)$ is monotonically increasing, $C_1(\theta)$ is monotonically decreasing,*
2. *$C_2(0.5) = 1+\epsilon$, $C_2(0.51) \leq 1.015$, $C_1(0,533) < 2$,*
3. *$C_1(0.5) = 1-\epsilon$, $C_1(0.52) > 0.16$.*

(Here by most *$q \leq x^\theta$ we mean that for all $g > 1$, the number of exceptional q is at most $x^\theta/(\log x)^g$, for $x \geq x_0(a,g)$. For our application we will take take $g = 2, a = 1$ and $\theta = 0.52$. Hence there is a fixed x_0 such that the number of exceptional q is at most $x^{0.52}/(\log x)^2$, for $x \geq x_0$.)*

[6] More precisely, this is an average version of the so called Brun-Titchmarsh Theorem.

Proof of Theorem 2.2.1

Using the Baker-Harman Theorem, we now present a proof of Theorem 2.2.1. Let $\theta = 0.52$. Consider all positive integers $q \leq x^{\theta}$ and $a = 1$. It follows from the Baker-Harman theorem, that for most $q \leq x^{\theta}$, $C_1(\theta) > 0$. This implies that for these q, $\pi(x; q, 1) > 0$, and hence at-least 1. This implies that for at-least $x^{0.52}/(\log x)^2$ positive integers q, which are less that x, the smallest prime in the arithmetic progression $\{kq + 1 | k \geq 0\}$ is at most x. Restricting ourselves to the case when q itself is a prime number, by Prime Number Theorem, we are still left with $2x^{0.52}/\log x - x^{0.52}/(\log x)^2$ *good* q-s. For large enough x this means that the number of primes $q \leq x^{0.52}$, such that the smallest prime in the arithmetic progression $\{kq + 1 | k \geq 0\}$ is at most x, is at-least $x^{0.52}/\log x$.

Recall that for a prime q, $p(q)$ denotes the smallest prime in the arithmetic progression $\{qk + 1 | k \geq 0\}$. It could be that for $q_1 \neq q_2$, $p(q_1) = p(q_2)$. We show that no more than 6 primes can share the same smallest prime. So if we take $6t$ distinct primes $\{q_i | 1 \leq i \leq 6t\}$, the cardinality of the set $|\{p(q_i) | 1 \leq i \leq 6t\}| \geq t$.

Lemma 2.2.9. *Let $q_0 < q_1 < q_2 < \cdots < q_v$ be primes such that the smallest prime in each of the arithmetic progression $\{jq_i + 1 | j \geq 1\}$ is p, and q_0 is the constant above which Linnik's Theorem applies. Then $v < L$.*

Proof. By hypothesis, there is an integer $k' > 1$ such that

$$p - 1 = k' q_1 q_2 \cdots q_v.$$

Hence $q_1^v \leq \prod_{i=1}^{v} q_i < p$. As p is the smallest prime in the arithmetic progression $\{kq_1 + 1 | k \geq 0\}$, we know from Linnik's Theorem that $p < q_1^L$. Hence $q_1^v < q_1^L$, implying the lemma. □

Let $\mathcal{P}_t := \left\{p | 1 \leq p \leq t^{1.924} \text{ and } p \text{ is a prime } \right\}$ and further define

$$\mathcal{Q}_t := \{p \in \mathcal{P}_t | p = p(q), \text{ for some prime } q\}.$$

By the discussion above, and the fact that $1.924 \times 0.52 = 1.0048 > 1$, Theorem 2.2.1 now follows. More precisely, there is a fixed constant t_0 such that $|\mathcal{Q}_t| \geq t$, for all $t \geq t_0$. This completes the proof of the theorem.

The important thing about Theorem 2.2.1 is that it is better than what one can hope even using the GRH. The catch is that using this hypothesis one can obtain (slightly inferior) bounds for **every** q. (Compare this to the bound of $t^{5.5}$ in [79]). But we have *unconditional* and *better* bounds.

Now we move on to the description of the algorithm and show how to use Theorem 2.2.1 to get Theorem 2.2.2.

2.2.4 An algorithm for testing polynomial identities

Transforming a multivariate polynomial to a univariate polynomial

First we need the following simple but useful transformation. Let $f(x_1, \ldots, x_n)$ be a polynomial over a ring of zero characteristic (e.g. $\mathbb{Z}$). Further, let the degree of f in each variable be bounded by d. (The maximum degree of f is bounded by d.) The following transformation maps such a multivariate polynomial f to a univariate polynomial g.

$$g_f(y) := f\left(y, y^{(d+1)}, y^{(d+1)^2}, \cdots, y^{(d+1)^{n-1}}\right).$$

Write $f = \sum_\alpha c_\alpha \mathbf{x}^\alpha$. By definition, $g_f(y) = \sum_\alpha c_\alpha g_{\mathbf{x}^\alpha}(y)$. Notice that for $\alpha \neq \beta$, $g_{\mathbf{x}^\alpha}(y) \neq g_{\mathbf{x}^\beta}(y)$. To see this, write $\alpha = (\alpha_1, \ldots, \alpha_n)$ and $\beta = (\beta_1, \ldots, \beta_n)$. $g_{\mathbf{x}^\alpha}(y)$ is $y^{\sum_{i=1}^n \alpha_i (d+1)^{i-1}}$ and $g_{\mathbf{x}^\beta}(y)$ is $y^{\sum_{i=1}^n \beta_i (d+1)^{i-1}}$. Since the maximum degree of f is at most d, each α_i and β_i are at most d. Also, no cancellation among monomial takes place as we are over a ring of zero characteristic. Thus, $g_{\mathbf{x}^\alpha}(y) = g_{\mathbf{x}^\beta}(y)$ if and only if $\alpha = \beta$. Hence we have proved the following lemma.

Lemma 2.2.10. *Given a polynomial $f(x_1, \ldots, x_n)$ over a characteristic zero field with maximum degree no more than d, the substitution $x_i \to x^{(d+1)^{i-1}}$ has the following properties:*

1. *f is identically zero if and only if g_f (defined above) is identically zero.*
2. *The degree of $g_f(x)$ is at most $n(d+1)^n$.*
3. *The number of non-zero monomials is the same in f and g_f. (*Sparsity *is preserved under this transformation.)*

The algorithm

Recall the definition of $\mathcal{P}_t$

$$\mathcal{P}_t := \left\{p | 1 \leq p \leq t^{1.924} \text{ and } p \text{ is a prime } \right\}$$

Input: f and integers $m \geq m(f), n, d \geq d(f), H \geq H(f)$

Compute $t = \lceil mn \log(d+1) + \log H \rceil + 1$

Compute the set $\mathcal{P}_t$

for $p \in \mathcal{P}_t$ **do**

  if $g_f(y) \not\equiv 0$ *on* GF_p **then**

    Output $f \not\equiv 0$

  end

end

Output $f \equiv 0$

Algorithm 1: Identity Testing

Note that since we assume an oracle access to f, we can implement the function g as follows: Once a prime p is fixed, and a point $a \in \mathsf{GF}_p$, we compute the transformation mentioned in Lemma 2.2.10 modulo p. Though the degree is exponential, exponentiation can be done fast by repeated squaring modulo p. Once we have the transformed input, we query f at that point. Then we take the output modulo p. Since the oracle of f is over a characteristic zero field, the modular arithmetic described is justified.

2.2.5 Analysis

Let $t := \lceil mn \log(d+1) + \log H \rceil + 1$, where m is an upper bound on the number of monomials in f, d is an upper bound on the maximum degree of f and H bounds the magnitude of the largest coefficient of f.

Assume on the contrary that $f \not\equiv 0$, but $g_f \equiv 0$ over GF_p, for all primes $p \in \mathcal{P}_t$. Recall that the set $\mathcal{Q}_t \subset \mathcal{P}_t$ is defined as:

$$\mathcal{Q}_t := \{p \in \mathcal{P}_t | p = p(q), \text{ for some prime } q\}.$$

We just focus on the primes in the set $\mathcal{Q}_t$. Let $g_f(y) = \sum_i c_i y^{k_i}$. Since $f \not\equiv 0$, there is a monomial, corresponding to lets say $\alpha = (\alpha_1, \ldots, \alpha_n)$, in f with $c_\alpha \neq 0$. Hence, the coefficient of $y^{k_0} := y^{\sum_{i=1}^{n} \alpha_i (d+1)^{i-1}}$ in $g_f(y)$ is non-zero. If f has at most m monomials, we already noted that $g_f(y)$ has at most m non-zero terms. Write $g_f(y)$ as $c_{k_0} y^{k_0} + \sum_{i=2}^{m} c_i y^{k_i}$. Call a prime $p \in \mathcal{Q}_t$ **bad** for f if at least one of the following occurs:

1. $p - 1 | (k_0 - k_j)$ for some $2 \leq j \leq m$.
2. p divides c_{k_0}.

Hence, if $\mathcal{Q}_t$ contains a prime p which is not bad for f, then the term $c_{k_0} y^{k_0}$ will not cancel with any other monomial that occurs in g_f, and itself will not end up being zero over GF_p. The following fact now follows immediately.

Fact 2.2.11. *The algorithm works correctly for all f, for which there is at-least one prime in the set $\mathcal{Q}_t$ which is not bad.*

We now turn to estimate the the number of bad primes for f. For the first case, let p be a prime such that $p - 1$ divides $k_0 - k_j$ for some and $2 \leq j \leq m$. Since $p \in \mathcal{Q}_t$, p is the smallest prime in the arithmetic progression $\{kq + 1 | k \geq 0\}$ for some prime q. Hence, q divides $p - 1$, and consequently q divides $k_0 - k_j$. For each $2 \leq j \leq k$, $k_0 - k_j$ has at most $\log |k_0 - k_j|$ distinct prime factors. Since the number of integers $|k_0 - k_j|$ is at most m and each of them could be at most $(d+1)^n$ in magnitude, the following can be deduced:

Fact 2.2.12. *By Lemma 2.2.3, the number of bad primes for the first case is at most $mn \log(d+1)$.*

The second case is straightforward. The number of bad primes that divide c_0 are at most $\log |c_0|$.

Fact 2.2.13. *By Lemma 2.2.3, the number of bad primes for the second case is at most*

$$\log |c_0| \leq \log H.$$

Hence, we complete the proof of the following theorem. As a corollary to it, one obtains Theorem 2.2.2.

Theorem 2.2.14. *The algorithm works correctly if*

$$t > (mn \log(d+1) + \log H).$$

2.3 Lower bounds

In this section we present known and new lower bounds for identity testing over various representations.

2.3.1 Circuits and determinants

Let $\mathcal{C}(n, d, c)$ be the set of circuits of size at most n^c, computing polynomials on n variables of maximum degree at most d (say over integers). The number of such circuits is at-most $2^{O(n^c \log n)}$. Hence applying Theorem 2.1.1, there is a test set of description length at most $O(dn^{c+1} \log n)$ which can be used to separate the zero polynomial from those which are not identically zero in $\mathcal{C}(n, d, c)$. Finding such a test set remains elusive and now with the results of Kabanets and Impagliazzo [54], seems hard. Similar existential (as determinant has small circuits) and hardness result (see [54]) hold for determinants too.

2.3.2 Black Box

The Black Box model is ubiquitous, most notably in Cryptographic applications, [22]. Also, this model is the most susceptible to lower bound arguments.

By a lower bound here one means the number of queries any deterministic algorithm can be forced to make, before it is certain that the box represents a zero polynomial. This bound translates easily into a randomized lower bound: the minimum number of random bits an algorithm needs to decide the same. The minimum number of random bits needed is roughly the logarithm of the minimum number of queries any deterministic algorithm needs to make, see [75].

Depending on how one uses the output, the lower bound argument changes. Also if it is a Promise Black Box model, one has to come up with more sophisticated arguments. Indeed if the black box is promised to be a circuit of small complexity, we cannot prove any thing non-trivial and probably no good lower bounds exist!

There are two primary ways in which the output of a black box has been used:

1. **Equality:** Evaluate f and test whether it is zero or not. This is used in almost all the algorithms.

2. **Approximate Equality:** Evaluate f and test if it is *close* to zero. Such a test was used by Chen and Kao [31].

We first give a general lower bound observation, first due to Lewin and Vadhan [75], and then go on to give near optimal lower bounds for promise and *approximate equality* tests.

Equality with promise on the number of monomials

The arguments in this section works over any field (or $\mathbb{Z}$). The following was observed in [79, 65].

Suppose we are given a $\mathsf{PBB}^f[0,(n,m,\cdot,\cdot)]$, i.e. either f is identically zero or has at most m monomials. We show that any deterministic algorithm can be forced to make m queries. This is because each query gives one linear relation in the coefficients of the unknown polynomial. So as long as the system is under determined, which happens when the algorithm makes less than m queries, since we are in a field, there with a non-zero solution to the coefficients which satisfies all the queries. Hence m queries are necessary.

Notice that this argument does not promise any interesting bounds on the coefficients.

Equality with multiple promises

Now we move to more refined estimates when the problem is of type $\mathsf{PBB}^f[0,(n,m,d,H)]$. But we restrict ourselves to $\mathbb{Z}$. Set $h := \log H$. The argument involved here is similar to those proving existence of short vectors in lattices.

Define $l' := \binom{n+\delta}{\delta}$ and $l := \binom{l'}{m}$. Let $\mathcal{F}(n,\delta,m,H)$ denote the family of polynomials $f(x_1,\ldots,x_n) \in \mathbb{Z}[x_1,\ldots,x_n]$ of total degree at most δ, with at most m monomials and height less than H. Then $|\mathcal{F}(n,\delta,m,H)| = \Theta\left(lH^m\right)$.

For the class of polynomials $\mathcal{F}(n,\delta,m,H)$, call a deterministic algorithm $\mathcal{A}_{t,Z}$ satisfying the following properties: *oblivious.*

Given $f(x_1,\ldots,x_n) \in \mathcal{F}(n,\delta,m,H)$, $\mathcal{A}_{t,Z}$ outputs $f \equiv 0$ if and only if for a set

of query points $\mathbf{q}_1, \ldots, \mathbf{q}_t$

$$f(\mathbf{q}_1) = 0, \ldots, f(\mathbf{q}_t) = 0.$$

Moreover each entry in each query vector $\mathbf{q}_i$ is at most Z in absolute value.

We say that $\mathcal{A}_{t,Z}$ *separates* $\mathcal{F}(n, \delta, m, H)$ if the only polynomial in $\mathcal{F}(n, \delta, m, H)$ that $\mathcal{A}_{t,Z}$ announces 0 is the 0 polynomial. Notice that if $\mathcal{A}_{t,Z}$ separates $\mathcal{F}(n, \delta, m, 2H)$, then $\mathcal{A}_{t,Z}$ separates $\mathcal{F}(n, \delta, m, H)$.

We have the following Theorem

Theorem 2.3.1. *There is an absolute constant $c > 0$, such that for any oblivious deterministic algorithm $\mathcal{A}_{t,Z}$ which separates $\mathcal{F}(n, \delta, m, 2H)$,*

$$t \geq c \cdot \frac{\log l + m \log H}{\log m + \log H + d \log Z}.$$

Proof. By remark above, $\mathcal{A}_{t,Z}$ also separates $\mathcal{F}(n, \delta, m, H)$. $\mathcal{A}_{t,Z}$ can be viewed as a mapping from $\mathcal{F}(n, \delta, m, H)$ to the space of t tuples. The size of the range is at most $(mHZ^d)^t$, as each coordinate is at most mHZ^d. If $|\mathcal{F}(n, \delta, m, H)| > (mHZ^d)^t$, then there are two (not both zero) polynomials $f_1, f_2 \in \mathcal{F}(n, \delta, m, H)$ which have the same image. Hence $f_1 - f_2$ is a non-zero polynomial in $\mathcal{F}(n, \delta, m, 2H)$ mapping to the t-tuple which is all zero. Hence contradicting the fact that $\mathcal{A}_{t,Z}$ separates $\mathcal{F}(n, \delta, m, 2H)$. Thus $\Omega(lH^m) \leq (mHZ^d)^t$. This completes the proof. □

It is interesting to note that as $d \to \infty$, the lower bound goes to zero. This seems to be an artifact of the proof rather than the problem. It would be interesting to remove this pathology from the bounds.

Approximate equality with multiple promises

In this section we establish a lower bound for black box identity testing algorithm of Chen and Kao [31]. For simplicity of presentation we assume that we are given a multilinear polynomial $f(x_1, \ldots, x_n) \in \mathbb{Z}[x_1, \ldots, x_n]$.

In their algorithm, Chen and Kao choose algebraic numbers $\alpha_1, \ldots, \alpha_n$ such that the degree of extension $[\mathbb{Q}(\alpha_1, \ldots, \alpha_n) : \mathbb{Q}] = 2^n$, then they observe:

$$f(\alpha_1, \ldots, \alpha_n) = 0 \text{ if and only if } f \equiv 0. \tag{1}$$

(For basic algebraic number theory refer to [91].) This follows from the *Theorem of Primitive Element* [91] of algebraic extensions. In particular the theorem says that there is an algebraic number θ such that $\mathbb{Q}(\theta) = \mathbb{Q}(\alpha_1, \ldots, \alpha_n)$. θ has 2^n conjugates: $\theta = \theta_1, \theta_2, \ldots, \theta_{2^n}$.

The trouble is that α_is are not rational, and we can evaluate f only at rational numbers. Hence they compute l bit rational approximations r_i of α_is. But now Equation (1) does not necessarily hold. The key idea is to choose a random sign vector $(s_1, \ldots, s_n) \in \{-1, 1\}^n$ and evaluate $f(s_1 \cdot r_1, \ldots, s_n \cdot r_n)$. (Algebraically this is same as choosing certain rational approximation to a random conjugate of θ.) They show using some basic properties of algebraic integers, that if $f \not\equiv 0$, then $|f(s_1 \cdot r_1, \ldots, s_n \cdot r_n)| \geq 2^{-l}$ with high probability. The remarkable fact about this argument is that the error probability can be reduced by increasing l and without using any more random bits!

We now give a lower bound argument on the random bits needed in this algorithm by arguing that any deterministic algorithm of this type has to evaluate the polynomial at many conjugates before it can be certain that f is identically zero or not.

First we make the notion of such an algorithm precise. Let $\mathcal{F}(n, H)$ denote the family of multilinear polynomials $f(x_1, \ldots, x_n) \in \mathbb{Z}[x_1, \ldots, x_n]$ with height less than H. (We omit the general case of d degree and m monomials.) $|\mathcal{F}(n, H)| = \Theta\left(H^{2^n}\right)$.

Let $\alpha_1, \ldots, \alpha_n$ be algebraic numbers such that $[\mathbb{Q}(\alpha_1, \ldots, \alpha_n) : \mathbb{Q}] = 2^n$. Moreover let $a := \max_i |\alpha_i|$. For the class of polynomials $\mathcal{F}(n, H)$, call a deterministic algorithm $\mathcal{A}_{t,a,l}$ satisfying the following properties: *oblivious approximator.*

> Given $f(x_1, \ldots, x_n) \in \mathcal{F}(n, H)$, $\mathcal{A}_{t,a,l}$ outputs $f \equiv 0$ if and only if for a set of query points $\mathbf{q}_1, \ldots, \mathbf{q}_t$
>
> $$|f(\mathbf{q}_1)| \geq 2^{-l}, \ldots, |f(\mathbf{q}_t)| \geq 2^{-l}.$$
>
> Each query vector $\mathbf{q}_i$ is of the form $(s_1 \cdot r_1, \ldots, s_n \cdot r_n)$, where $|r_i - \alpha_i| \leq 2^{-l}$ and $s_i \in \{-1, 1\}$ for all $1 \leq i \leq n$.

We say that $\mathcal{A}_{t,a,l}$ *separates* $\mathcal{F}(n,H)$ if the only polynomial in $\mathcal{F}(n,H)$ that $\mathcal{A}_{t,a,l}$ announces 0 is the 0 polynomial.

For simplicity we assume that $r_i = \alpha_i$. The effect of the approximation is very little and conceptually does not any difference. The proof uses a volume argument. We have the following Theorem

Theorem 2.3.2. *There is an absolute constant $c > 0$, such that for any oblivious approximator deterministic algorithm $\mathcal{A}_{t,a,l}$ which separates $\mathcal{F}(n,2H)$,*

$$t \geq c \cdot \frac{2^n \log H}{2l + \log H + n \log a + n}.$$

Proof. Suppose there is such an algorithm $\mathcal{A}_{t,a,l}$ which separates $\mathcal{F}(n,2H)$, then it also separates $\mathcal{F}(n,H)$. $\mathcal{A}_{t,a,l}$ can be thought of as a mapping from $\mathcal{F}(n,H)$ to the space of t tuples. the magnitude of each coordinate can be at most $Ha^n 2^n$. Hence the image of $\mathcal{F}(n,H)$ sits in a volume of magnitude at most $(Ha^n 2^n)^t$. If $|\mathcal{F}(n,H)| > 2^{-2lt}(Ha^n 2^n)^t$, then there are two (not both zero) polynomials $f_1, f_2 \in \mathcal{F}(n,H)$ whose images lie in a t dimensional cube of volume of side at most 2^{-2l}. Hence in absolute value, each coordinate of the image of $f_1 - f_2$ (which is a non-zero polynomial) is less than 2^{-l} in absolute value. Hence $\mathcal{A}_{t,a,l}$ will not be able to distinguish it from the zero polynomial. Moreover the coefficients of $f_1 - f_2$ are at most $2H$. Thus a contradiction is achieved. □

CHAPTER III

HILBERT'S 17TH PROBLEM

3.1 Introduction

Hilbert proposed 23 problems in 1900, in which he tried *to lift the veil behind which the future lies hidden.*[1] His description of the 17th problem is (see [1]):

A rational integral function or form in any number of variables with real coefficient such that it becomes negative for no real values of these variables, is said to be definite. The system of all definite forms is invariant with respect to the operations of addition and multiplication, but the quotient of two definite forms in case it should be an integral function of the variables is also a definite form. The square of any form is evidently always a definite form. But since, as I have shown [46], not every definite form can be compounded by addition from squares of forms, the question arises which I have answered affirmatively for ternary forms [47] whether every definite form may not be expressed as a quotient of sums of squares of forms. At the same time it is desirable, for certain questions as to the possibility of certain geometrical constructions, to know whether the coefficients of the forms to be used in the expression may always be taken from the realm of rationality given by the coefficients of the form represented.

An affirmative answer to this problem was given by Emil Artin in 1927 [9]:

> For every non-negative polynomial $f \in \mathbb{R}[x_1, \ldots, x_n]$, there exist rational functions $g_1, \ldots, g_s \in \mathbb{R}(x_1, \ldots, x_n)$, such that $f = g_1^2 + \cdots + g_s^2$.

Motzkin's example (see [105]) of $P(x, y, z) = z^6 + x^4z^2 + x^2y^4 - 3x^2y^2z^2$ illustrates that the rational functions in Artin's result cannot, in general, be replaced by polynomials. $P(x, y, z)$ is non-negative everywhere over the reals, and yet, cannot be written as sum of

[1] A quote taken from [114].

squares of polynomials over the reals. Notice that Artin's result shows that every non-negative polynomial can be written as sum of squares of *finitely* many rational functions. This raised the following important question about the *size* of such a representation:

> What is the smallest number (denoted as $\nu(n,d)$), such that every n-variate, degree d, non-negative polynomial can be written as sum of squares of $\nu(n,d)$ rational functions over the reals?

In 1967, Pfister [96] proved that $\nu(n,d) \leq 2^n$. However, this upper bound holds when one is allowed rational functions over a *real closed field* [2]. Remarkably enough, his bound does not depend on the degree of the polynomial. The best lower bound on $\nu(n,3)$ is $n+2$. Over 75 years of effort by various mathematicians, these are still the best known bounds in general. We remark that the function $\nu(n,2)$ is quite well understood from the time of Hilbert (see [46, 48, 49]).

In lieu of the lack of progress towards the determination of $\nu(n,d)$, we initiate the study of Hilbert's 17th problem from the point of view of Computational Complexity. In this setting, the following question is a natural relaxation:

> What is the **descriptive complexity** of the sum of squares representation (as rational functions) of a non-negative, n-variate, degree d polynomial?

We consider *arithmetic circuits* as a natural representation of rational functions. We are able to show, assuming a standard conjecture in complexity theory, that it is impossible that every non-negative, n-variate, degree four polynomial can be represented as a sum of squares of a *small* (polynomial in n) number of rational functions, each of which has a *small* size arithmetic circuit (over the rationals) computing it.

Our result points to the direction that it is unlikely that every non-negative, n-variate polynomial over the reals can be written as a sum of squares of a polynomial (in n) number of rational functions. Further, relating to standard (and believed to be hard to prove)

[2] See [21, 98] for a definition.

complexity-theoretic conjectures sheds some light on why it has been difficult for mathematicians to close the $n+2$ and 2^n gap.

3.1.1 Related work

Like all of Hilbert's problems, the 17th has received a lot of attention from the mathematical community and beyond. For an extensive survey of the development and impact of Hilbert's 17th problem on Mathematics, the reader is referred to excellent surveys by [41, 101, 105, 113]. The books [21, 98] also provide good accounts of this and related problems.

Apart from what can be found in the references above, we are aware of some recent work on various quantitative aspects of Hilbert's 17th problem. For instance, in [14], it has been proved that if the degree is fixed and the number of variables are allowed to increase, then there are significantly many more non-negative polynomials than those that can be written as sum of squares of polynomials. Further, in [102], it is shown that in general, one cannot obtain a sum of squares representation in which each rational function has the *same* denominator.

To the best of our knowledge the problem raised by this work, about the representational complexity of non-negative polynomials in the computational setting, is new.

3.2 Overview of our result

Notations

For $k = \mathbb{R}, \mathbb{Q}$ or $\mathbb{Z}$, $k[x_1, \ldots, x_n]$ denotes the ring of polynomials over k and $k(x_1, \ldots, x_n)$ denotes the corresponding field of fractions. The following notation about polynomials is used throughout this chapter: A polynomial is written as $f = \sum_\alpha c_\alpha \mathbf{x}^\alpha$. Here $\mathbf{x}^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$. $\deg(f)$ denotes the maximum *total degree* of f. $H(f) := \max_\alpha |c_\alpha|$.

Arithmetic circuits

An *arithmetic circuit* C over k [3] is a directed acyclic graph. Each vertex has indegree 0 or 2 and is labeled either by addition, multiplication, one of the input variables: $\{x_1, \ldots, x_n\}$,

[3] In general k could be a commutative ring, but here k will be either the fields $\mathbb{R}$ and $\mathbb{Q}$, or the ring of integers $\mathbb{Z}$.

or scalars from k. If the vertex is labeled by a scalar or an input variable, then its indegree must be 0. If the vertex has indegree 2, then it must be labeled either by $+$ or by $\times$. There is exactly one vertex with no outgoing edge, which naturally corresponds to the polynomial (over $k[x_1, \ldots, x_n]$) computed by C. The size of C is the number of gates along with description size of all the constants used. As observed, C computes a polynomial $f(x_1, \ldots, x_n) \in k[x_1, \ldots, x_n]$. The size of the smallest arithmetic circuit that computes $f \in k[x_1, \ldots, x_n]$ is denoted by $\mathcal{L}_k(f)$. We will drop the subscript wherever k is clear from the context.

Unsatisfiability

Consider a boolean function $\phi : \{0,1\}^n \mapsto \{0,1\}$ in the conjunctive normal form (3-CNF), that is $\phi(x_1, \ldots, x_n) = \bigwedge_{i=1}^{m} C_i$, where each C_i is a boolean OR of at most 3 literals from $\{x_1, \overline{x}_1, \ldots, x_n, \overline{x}_n\}$. ϕ is said to be *satisfiable* if there is a *satisfying assignment* $a_1, \ldots, a_n \in \{0,1\}$, such that $\phi(a_1, \ldots, a_n) = 1$. The set of such boolean functions, in 3-CNF form, that have a satisfying assignment is denoted $\mathsf{3SAT}$. It is well known that $\mathsf{3SAT}$ is NP-complete. The corresponding $\mathsf{co\text{-}NP}$ problem is $\mathsf{UN3SAT}$, i.e. the set of boolean functions in 3-CNF that have no satisfying assignment. It follows that $\mathsf{UN3SAT}$ is complete for $\mathsf{co\text{-}NP}$.

Now we give the key definition and the main result of this chapter. (Readers not familiar with standard notations and definitions in computational complexity theory, should refer to the appendix.)

Definition 3.2.1.

$$\mathsf{H}^{\mathbb{Z}}(n, d, h) := \{f \in \mathbb{Z}[x_1, \ldots, x_n] : \deg(f) \leq d, H(f) = O(h), \forall (x_1, \ldots, x_n) \in \mathbb{R}^n \; f \geq 0\}.$$

Further, let $\mathsf{H}^{\mathbb{Z}}(d, h) := \cup_{n \geq 0} \mathsf{H}^{\mathbb{Z}}(n, d, h)$.

Remark 3.2.2. *Note that we are implicitly viewing* $\mathsf{H}^{\mathbb{Z}}(d, h)$ *as a language. Fixing a unique representation of polynomials (say the smallest arithmetic circuit over* $\mathbb{Q}$*), we can view polynomials in this set as binary strings, thus, justifying our viewpoint. Hence, the length of the input is related to the description of the polynomial and* **not** n*. But we concern*

ourselves only with the case when the smallest arithmetic circuit computing an n-variate polynomial f is of size at most a fixed polynomial in n, say n^6. [4]

3.2.1 Main theorems

Theorem 3.2.3. *Assuming* $\mathsf{PH} \neq \Sigma_2$*, for all* $n \geq 1$*, there exists a polynomial* $f \in \mathsf{H}^{\mathbb{Z}}(n, 6, 1)$ *such that no representation of* f *as sum of squares of rational functions over* $\mathbb{Q}$*,* $f = \sum_{i=1}^{s} g_i^2$*,* $g_i \in \mathbb{Q}(x_1, \ldots, x_n)$*, satisfies both of the following:*

1. $s = \text{poly}(\mathcal{L}(f))$.
2. *For all* $i = 1, 2, \ldots, s$, $\mathcal{L}(g_i) = \text{poly}(\mathcal{L}(f))$.

Thus, unless the polynomial hierarchy collapses to the second level, not every non-negative polynomial has a *succinct* sum of squares representation. It is a standard hypothesis in complexity theory that $\mathsf{PH} \neq \Sigma_2$.[5] In fact this theorem says that even if the polynomial has degree 6 and all coefficients are integers and bounded by a constant, there is no such representation. As remarked earlier, the degree 2 case is well understood. We strengthen the previous result by bringing the degree down to 4, at the cost of blowing up the size of the coefficients. It is an interesting open problem if such a statement can be obtained for degree 3.

Theorem 3.2.4. *Assuming* $\mathsf{PH} \neq \Sigma_2$*, for all* $n \geq 1$*, there is a polynomial* $f \in \mathsf{H}^{\mathbb{Z}}(n, 4, \text{poly}(n))$ *such that no representation of* f *as sum of squares of rational functions over the rationals,* $f = \sum_{i=1}^{s} g_i^2$*,* $g_i \in \mathbb{Q}(x_1, \ldots, x_n)$*, satisfies both of the following:*

1. $s = \text{poly}(\mathcal{L}(f))$.
2. *For all* $i = 1, 2, \ldots, s$, $\mathcal{L}(g_i) = \text{poly}(\mathcal{L}(f))$.

[4] For a non-negative, n-variate polynomial with arithmetic circuit complexity not bounded by any polynomial in n, one cannot hope to write an efficient (polynomial in n) sum of square representation by rational functions. Hence it makes sense only to consider polynomials which are efficiently computable by small circuits.

[5] Refer to the appendix for a substantiation of this belief.

A remark about the representation field

Although we state our theorems for $\mathbb{Q}$, one can replace it by a finite real algebraic extension of $\mathbb{Q}$. The details are easy and we omit the details for the ease of presentation. It is important to note though, that Artin's result does not, in general, imply existence of a sum of squares representation, where each rational function is over $\mathbb{Q}$. The *hard to represent* polynomials guaranteed by our results have a further property that these have small arithmetic circuits over the integers. It is conceivable that for such polynomials, a succinct representation (in our sense) exists if and only if a succinct representation exists over the reals. This is an interesting question for which we do not know an answer.

Outline of the proofs

As the first step in the proof of Theorems 3.2.3 and 3.2.4, we reduce an instance ϕ of $\mathsf{UN3SAT}$ to a polynomial F_ϕ which is non-negative if and only if ϕ is unsatisfiable. This is a variant of an often used trick, which allows one to use algebraic considerations to study a boolean formula. We give two such reductions, corresponding to the two theorems: for Theorem 3.2.3 we give a reduction such that $F_\phi \in \mathsf{H}^{\mathbb{Z}}(6,1)$ and for Theorem 3.2.4, $F_\phi \in \mathsf{H}^{\mathbb{Z}}(4,\mathrm{poly}(\cdot))$. These results establish the $\mathsf{co\text{-}NP}$ hardness of the classes $\mathsf{H}^{\mathbb{Z}}(6,1)$ and $\mathsf{H}^{\mathbb{Z}}(4,\mathrm{poly}(\cdot))$. Artin's Theorem guarantees a sum of squares representation of F_ϕ over the reals. If there is some such representation which is *succinct* (describable by a polynomial number of polynomial size arithmetic circuits), in NP we can guess it and in $\mathsf{co\text{-}RP}$, check if the guessed representation is the same as F_ϕ. (This last step is done by invoking polynomial identity testing.) Formally, we prove the following theorem:

Theorem 3.2.5. *For all $n, d, h \geq 1$, if for all $f \in \mathsf{H}^{\mathbb{Z}}(n,d,h)$, there exist $g_1, g_2, \ldots, g_s \in \mathbb{Q}(x_1, \ldots, x_n)$, such that $f = \sum_{i=1}^{s} g_i^2$, $s = \mathrm{poly}\left(\mathcal{L}(f)\right)$, and for all $i = 1, 2, \ldots, s$, $\mathcal{L}(g_i) = \mathrm{poly}\left(\mathcal{L}(f)\right)$, then $\mathsf{H}^{\mathbb{Z}}(d,h) \in \mathsf{NP}^{\mathsf{co-RP}}$.*

To derive the desired contradiction, in the end we invoke a result of Boppana, Hastad and Zachos [23], which states that $\mathsf{co\text{-}NP} \not\subseteq \mathsf{NP}^{\mathsf{co-RP}}$, unless $\mathsf{PH} = \Sigma_2$.

Organization

Section 3.3 contains the arithmetizations of SAT needed to prove Theorems 3.2.3 and 3.2.4. In Section 3.4.1, we define the problem of identity testing. This will be useful in the proof of the main result in Section 3.4, which contains proofs of Theorems 3.2.3, 3.2.4, 3.2.5.

3.3 Arithmetization of SAT

In this section we give two different arithmetizations of instances of UN3SAT, each of which will be used in proving one of Theorems 3.2.3, 3.2.4.

Given an instance $\phi = \wedge_{i=1}^{m} C_i$ of a UN3SAT problem: Call a literal $z \in \{z_1, \overline{z}_1, \ldots, z_n, \overline{z}_n\}$ *positive*, if $z \in \{z_1, \ldots, z_n\}$. Else, call it *negative*. For a clause $C = C_+ \vee C_-$ (C_+ consists of positive literals while C_- consists of negative literals), define

$$\mathcal{A}(C) := \left(\prod_{z \in C_+} (1 - z) \right) \cdot \left(\prod_{z \in C_-} z \right).$$

For instance, if $C = x_1 \vee \overline{x}_2 \vee x_3$, then $\mathcal{A}(C) = (1 - x_1)x_2(1 - x_3)$. Further for $a_1, a_2, a_3 \in \{0, 1\}$, $\mathcal{A}(C)(a_1, a_2, a_3) = 0$ if and only if $C(a_1, a_2, a_3) = 1$, (or C is satisfiable). Now define

$$F_\phi(z_1, \ldots, z_n) := 300 \left(\sum_{i=1}^{n} z_i^2 (1 - z_i)^2 + \sum_{j=1}^{m} (\mathcal{A}(C_j))^2 \right) - 1. \tag{2}$$

Thus for all ϕ, $F_\phi \in \mathbb{Z}[z_1, \ldots, z_n]$. It is convenient to let $f_\phi := F_\phi/300$. The problem remains the same though, as the sign of f_ϕ is the same as that of F_ϕ. Let $\epsilon = \frac{1}{300}$.

Lemma 3.3.1. *ϕ is not satisfiable if and only if $f_\phi \geq 0$ over the reals.*

Proof. If ϕ is satisfiable, let $a = (a_1, \ldots, a_n) \in \{0, 1\}^n \subset \mathbb{R}^n$ be a satisfying assignment. Then by definition $f_\phi(a) = -\epsilon < 0$. To prove the converse, consider the case when ϕ is unsatisfiable. We need to show that $f_\phi \geq 0$ over the reals. Let $\delta = 1/4$. We consider two cases:

1. Case 1: Let $(s_1, \ldots, s_n) \in \mathbb{R}^n$ be a point such that there is an $1 \leq i \leq n$ such that s_i does not lie in either of the two intervals: $[-\delta, \delta], [1 - \delta, 1 + \delta]$. In this case $s_i^2(1 - s_i)^2 > \delta^4$. Since $\epsilon \leq \delta^4$, $f_\phi(s_1, \ldots, s_n) > 0$.

2. Case 2: Hence, we may assume that for a point $(s_1, \ldots, s_n)$, all s_i are in one of the intervals: $[-\delta, \delta], [1-\delta, 1+\delta]$. From this we construct a point $a = (a_1, \ldots, a_n) \in \{0,1\}^n$ as follows:

   - If $s_i \in [-\delta, \delta]$ then let $a_i = 0$.
   - If $s_i \in [1-\delta, 1+\delta]$ then let $a_i = 1$.

   Since ϕ is unsatisfiable, there is a clause, say C, which is not satisfied by a. Let $\mathcal{A}(C) = \left(\prod_{z \in C_+}(1-z)\right) \cdot \left(\prod_{z \in C_-} z\right)$. If $z_i \in C_+$, since C is not satisfied by a, it must be that $a_i = 0$, and hence $s_i \in [-\delta, \delta]$, or equivalently $(1-s_i) \in [1-\delta, 1+\delta]$. Similarly, if $\overline{z}_i \in C_-$, $a_i = 1$, and hence $s_i \in [1-\delta, 1+\delta]$. This implies that at the point $(s_1, \ldots, s_n)$, $f_\phi \geq \mathcal{A}^2(C) \geq (1-\delta)^6 > \epsilon$.

Thus, if ϕ is unsatisfiable, $f_\phi > 0$ over the reals. This completes the proof. □

The above arithmetization reduces UN3SAT to $\mathsf{H}^{\mathbb{Z}}(6,1)$. Thus, the following proposition follows from Lemma 3.3.1 and co-NP hardness of UN3SAT.

Proposition 3.3.2. $\mathsf{H}^{\mathbb{Z}}(6,1)$ *is* co-NP *hard.*

Next we show how to obtain a quantitatively better result, if we allow the coefficients to grow with the input size. First, we need a new reduction. As before, let ϕ be a boolean function given in 3-CNF form on n variables and m clauses.

$$f'_\phi(z_1, \ldots, z_n) := \sum_{i=1}^{n} \frac{(3^3+1)m}{\delta(m)^4} z_i^2 (1-z_i)^2 + \sum_{j=1}^{m} (\mathcal{A}(C_j)) - \epsilon(m). \tag{3}$$

Here δ and ϵ are positive functions (but less than 1) of m such that $\epsilon < (1-\delta)^3 - m\delta(1+\delta)^2$. Note that one can choose such a δ and an ϵ since $(1-\delta)^3 \to 1$ and $m\delta(1+\delta)^2 \to 0$ as $\delta \to 0$. As in the previous case, we can always multiply f'_ϕ suitably to obtain a polynomial F'_ϕ over the integers.

Lemma 3.3.3. *ϕ is not satisfiable if and only if $f'_\phi \geq 0$ over the reals.*

Proof. If ϕ is satisfiable, let $a = (a_1, \ldots, a_n) \in \{0,1\}^n \subset \mathbb{R}^n$ be a satisfying assignment. Then by definition $f'_\phi(a) = -\epsilon < 0$. To prove the converse, consider the case when ϕ is unsatisfiable. We need to show that $f'_\phi \geq 0$ over the reals. We consider two cases:

Case 1: Suppose that for a point $s := (s_1, \ldots, s_n)$, all s_i are in one of the intervals: $[-\delta, \delta], [1-\delta, 1+\delta]$. From this we construct a point $a = (a_1, \ldots, a_n) \in \{0,1\}^n$ as follows:

- If $s_i \in [-\delta, \delta]$ then let $a_i = 0$.
- If $s_i \in [1-\delta, 1+\delta]$ then let $a_i = 1$.

Since ϕ is unsatisfiable, there is a clause, say C, which is not satisfied by a. Let $\mathcal{A}(C) = \left(\prod_{z \in C_+}(1-z)\right) \cdot \left(\prod_{z \in C_-} z\right)$. If $z_i \in C_+$, since C is not satisfied by a, $s_i \in [-\delta, \delta]$, or equivalently $(1-s_i) \in [1-\delta, 1+\delta]$. Similarly, if $\overline{z}_i \in C_-$, $a_i = 1$ and hence $s_i \in [1-\delta, 1+\delta]$. This means that at the point s, $\mathcal{A}(C) \geq (1-\delta)^3$.

Now consider a clause C' satisfied by a. Writing $C' = C'_+ \bigvee C'_-$, we see that either some variable in C'_+ is set to 1, or some variable in C'_- is set to 0 in the assignment a. Without loss of generality, assume that $z_i \in C'_+$ is set to 1 ($a_i = 1$). Thus, $s_i \in [1-\delta, 1+\delta]$, or $(1-s_i) \in [-\delta, \delta]$. Thus

$$\mathcal{A}(C') = \left(\prod_{z \in C'_+}(1-z)\right) \cdot \left(\prod_{z \in C'_-} z\right) \geq -\delta(1+\delta)^2.$$

Adding the inequalities for unsatisfied and satisfied clauses, one gets that

$$\sum_{j=1}^{m} \mathcal{A}(C_j) \geq (1-\delta)^3 - m\delta(1+\delta)^2.$$

By the choice of ϵ and δ, we have $\epsilon < (1-\delta)^3 - m\delta(1+\delta)^2$, and therefore $f'_\phi(s) > 0$.

Case 2: Now consider a point $s = (s_1, \ldots, s_n)$ such that, there is an $1 \leq i \leq n$, such that s_i does not lie in either of the two intervals: $[-\delta, \delta], [1-\delta, 1+\delta]$. For a clause C, define

$$\Delta_C := \max\left\{\{|1-s_i| : z_i \in C_+\} \cup \{|s_j| : \overline{z}_j \in C_-\}\right\}.$$

It follows that $\mathcal{A}(C)(s) \geq -\Delta_C^3$. Now consider the following 2 cases:

Case 2a $\Delta_C > 3$

Let s_{j^*} be such that either $|s_{j^*}|$ or $|1-s_{j^*}|$ is equal to Δ_C. Then $(s_{j^*})^2(1-s_{j^*})^2 \geq \Delta_C^2(\Delta_C - 1)^2 > \Delta_C^3 + 1$. This implies that $\mathcal{A}(C)(s) + \frac{3^3+1}{\delta^4}(s_{j^*})^2(1 - s_{j^*})^2 > -\Delta_C^3 + \frac{3^3+1}{\delta^4}(\Delta_C^3 + 1) > 1$. The last inequality follows by noticing that $\delta < 1$.

Case 2b $\Delta_C \leq 3$

From the definition of case 2, $\exists s_{j^*}$ such that $(s_{j^*})^2(1 - s_{j^*})^2 \geq \delta^4$. Hence, $\frac{3^3+1}{\delta^4}(s_{j^*})^2(1 - s_{j^*})^2 > 3^3 + 1$. By definition of Δ_C, $\mathcal{A}(C)(s) \geq -3^3$. Combining these inequalities, we get $\mathcal{A}(C)(s) + \frac{3^3+1}{\delta^4}(s_{j^*})^2(1 - s_{j^*})^2 > 1$.

Now summing over all clauses, we get, $\sum_{j=1}^{m} \mathcal{A}(C_j)(s) + \sum_{i=1}^{n} \frac{(3^3+1)m}{\delta^4} s_i^2(1-s_i)^2 > m$. This is exactly what we set out to prove: $f_\phi(s) > 0$.

Thus, if ϕ is unsatisfiable, $f'_\phi > 0$ over the reals. This completes the proof. □

This leads to the following:

Proposition 3.3.4. $\mathsf{H}^{\mathbb{Z}}(4, \mathrm{poly}(\cdot))$ *is* co-NP *hard.*

Amplifying positivity

Using the PCP Theorem of [8], one can transform the given formula so that, if it is unsatisfiable, then a large fraction (say $c, 0 < c < 1$) of clauses are unsatisfiable. This gives rise to an arithmetization such that $f_\phi > cm - 1$ if and only if ϕ is unsatisfiable. This shows that even if one is given that whenever $f > 0$, $f > cm - 1$, it is still co-NP hard to decide the positivity of f.

Circuit complexity of the arithmetized polynomials

It is important to note that for any 3CNF formula ϕ, there is an arithmetic circuit over $\mathbb{Z}$ which computes F_ϕ and F'_ϕ, whose sizes are at most n^6. [6] In fact, the explicit arithmetizations written down earlier can be converted into such circuits.

[6] Since ϕ is in 3CNF, $m \leq (2n)^3$.

3.4 Main results

3.4.1 Testing identities

The *Identity Testing* problem for arithmetic circuits is to decide if two given arithmetic circuits evaluate the same polynomial. More formally, given two arithmetic circuits C_1, C_2 over $\mathbb{Z}$, let $f, g \in \mathbb{Z}[x_1, \ldots, x_n]$ be the polynomials computed by them respectively. The problem is to decide efficiently if $f - g$ is identically zero over the integers. Here, efficiency is measured in terms of the input size, which in this case, is the sum of the sizes of C_1 and C_2. The following result by Ibarra and Moran [51] establishes that, in the presence of randomness, there is an efficient solution to this problem. Formally, there is an efficient randomized algorithm which takes as input two circuits and decides if they compute the same polynomial. The algorithm is always correct when it says NO, but there is a small chance that is is wrong when it says YES. This simple but important result will play a crucial role in the proof of the main results which we describe next.

Lemma 3.4.1. *([51]) The Identity Testing problem for arithmetic circuits over $\mathbb{Z}$ is in* co-RP.

3.4.2 Proof of main theorems

The fact that non-negative polynomials can be represented as sum of squares suggests the following algorithm for checking if $f \in \mathsf{H}^{\mathbb{Z}}(n, d, \cdot)$. Suppose it is true that $f = g_1^2 + \cdots + g_s^2$, where each $g_i = \frac{\alpha_i}{\beta_i}$, α_i and β_i are polynomials over the integers, and $\beta_i \neq 0$. Further assume that this representation is *succinct*, that is $s = \text{poly}(n)$ and for all $1 \leq i \leq s$, $\mathcal{L}(\alpha_i), \mathcal{L}(\beta_i) \leq \text{poly}(n)$. Then in NP, we can *guess* these polynomials α_i, β_i, as the total bits one has to guess is a polynomial in n. Once we have guessed the representation, one checks the following identity:

$$f \prod_{j=1}^{s} \beta_j^2 - \sum_{i=1}^{s} \left(\alpha_i \prod_{j \neq i} \beta_j \right)^2 \equiv 0 \tag{4}$$

Since f itself has an arithmetic circuit over the integers of size at most n^6, the polynomial on the LHS of the above identity has a polynomial size circuit. Hence using the identity testing algorithm for arithmetic circuits over the integers, one can verify the above identity

in co-RP. Thus checking the validity of the guessed representation.

This is formalized in the following proof:

Proof of Theorem 3.2.5. Using NP, guess each $g_i = \frac{\alpha_i}{\beta_i}$ where α_i and β_i are polynomials over $\mathbb{Q}$. By hypothesis, we know that $f\prod_{j=1}^{s}\beta_j^2$ and $\sum_{i=1}^{s}\left(\alpha_i\prod_{j\neq i}\beta_j\right)^2$ are arithmetic circuits with length a polynomial in n. Hence by Lemma 3.4.1, checking whether they are equal is in co-RP. The time required to evaluate the g_i's is also a polynomial in n. Hence we get $\mathsf{H}^{\mathbb{Z}}(d,h) \in \mathsf{NP}^{\mathsf{co-RP}}$.

□

Finally, we need the following result of Bopanna, *et al* [23].

Theorem 3.4.2. *[23]* $\mathsf{co\text{-}NP} \subseteq \mathsf{NP}^{\mathsf{co-RP}} \Rightarrow \mathsf{PH} = \Sigma_2$.

Now we are ready to prove Theorem 3.2.3.

Proof of Theorem 3.2.3. Assume on the contrary. From Theorem 3.2.5, $\mathsf{H}^{\mathbb{Z}}(6,1) \in \mathsf{NP}^{\mathsf{co-RP}}$. But $\mathsf{H}^{\mathbb{Z}}(6,1)$ is co-NP-Hard by Proposition 3.3.2. Now by Theorem 3.4.2, $\mathsf{PH} = \Sigma_2$, a contradiction is achieved. □

Using Proposition 3.3.4 instead of Proposition 3.3.2 in the above proof, one obtains a proof of Theorem 3.2.4.

3.5 Appendix: Computational Complexity preliminaries

The aim of this section is to present the definitions and notions in Computational Complexity Theory. [7] The reader is refered to the book by Papadimitriou [95] for a comprehensive treatment of this subject.

Some complexity classes

A *language* is a subset of $\{0,1\}^*$. For a language L, $\bar{L} := \{0,1\}^*\backslash L$. A p-ary *relation* is a language over the following p-ary product: $\{0,1\}^* \times \cdots \times \{0,1\}^*$.[8] The complexity class

[7] The reason we do so is it to broaden the scope of this paper to mathematicians who may not be familiar with these notions, but are interested in understanding our results on Hilbert's 17th problem.

[8] A 1-ary relation is just a language.

$\mathsf{DTIME}(f(n))$ is the set of all languages for which membership can be tested in time $f(n)$, by a deterministic Turing machine, in time $f(n)$. $\mathsf{P} := \cup_{t \geq 0} \mathsf{DTIME}(n^t)$. NP is the collection of all languages L, such that there is a 2-ary relation $R_L \in \mathsf{P}$ (called a *polynomially decidable relation*) and a polynomial $p(\cdot)$, such that $x \in L$ if and only if there is a $y \in \{0,1\}^*$, with $|y| = O(p(|x|))$, and $(x,y) \in R_L$. The class $\mathsf{co\text{-}NP}$ is defined as $\cup_{L \in \mathsf{NP}} \bar{L}$. It follows that a language L is in $\mathsf{co\text{-}NP}$ if and only if there is a polynomially decidable 2-ary relation R_L and a polynomial $p(\cdot)$, such that $x \in L$ if and only if and for all $y \in \{0,1\}^*$, with $|y| = O(p(|x|))$, $(x,y) \in R_L$. It is natural to define complexity classes based on compositions of these *existential* and *universal* quantifiers. Starting with $\Sigma_1 = \mathsf{NP}$ and $\Pi_1 = \mathsf{co\text{-}NP}$, one can define Σ_i and Π_i as follows. For $i \geq 2$, Σ_i is the collection of all languages L such that there is a i-ary relation $R_L \in \Pi_{i-1}$, and a polynomial $p(\cdot)$, such that $x \in L$ if and only if there exists a $y \in \{0,1\}^*$, with $|y| = O(p(|x|))$, $(x,y) \in R_L$. Π_i is defined similarly as $\mathsf{co\text{-}}\Sigma_i$. Further, define $\Delta_i := \Sigma_i \cap \Pi_i$. One often thinks of $\Delta_0 = \Sigma_0 = \Pi_0 = \mathsf{P}$ and $\Delta_1 = \mathsf{NP} \cap \mathsf{co\text{-}NP}$. *Polynomial Hierarchy* (PH) is defined to be the collection of classes Δ_i, Σ_i and Π_i, for all $i \geq 0$. It follows from definitions that if $\mathsf{NP} = \mathsf{co\text{-}NP}$ then $\Sigma_i = \Delta_i$ for all $i \geq 1$.

Completeness

A language L is said to be *hard* for a a complexity class $\mathcal{C}$, for all $L' \in \mathcal{C}$, there is a polynomial $p(\cdot)$ and a Turing machine $M_{L,L'} : \{0,1\}^* \to \{0,1\}^*$, such that $x \in L$ if and only if $M_{L,L'}(x) \in L'$. Moreover, for the complexity classes we will be interested in, we assume that $M_{L,L'}$ runs in time $O(p(|x|))$. If $L \in \mathcal{C}$ and L is hard for $\mathcal{C}$, then L is said to be *complete* for $\mathcal{C}$. Complete problems for a complexity class can be thought of as the hardest problems in their class and can be thought of as characterizing the complexity class.

Next we define a problem which is known to be NP-complete. Consider a boolean function $\phi : \{0,1\}^n \mapsto \{0,1\}$ in the conjunctive normal form (3-CNF), that is $\phi(x_1, \ldots, x_n) = \bigwedge_{i=1}^{m} C_i$, where each C_i is a boolean OR of at most 3 literals from $\{x_1, \overline{x}_1, \ldots, x_n, \overline{x}_n\}$. ϕ is said to be *satisfiable* if there is a *satisfying assignment* $a_1, \ldots, a_n \in \{0,1\}$, such that $\phi(a_1, \ldots, a_n) = 1$. The set of such boolean functions, in 3-CNF form, that have a satisfying assignment is denoted $\mathsf{3SAT}$. One of the earliest and most important results in complexity

Theory (see [35, 59, 74]) was establishing that 3SAT is NP-complete. The corresponding co-NP problem is UN3SAT, i.e. the set of boolean functions in 3-CNF that have no satisfying assignment. It follows that UN3SAT is complete for co-NP. Generalizing these results, it is known that there is a complete problem for Σ_i (and hence for each Π_i), for all $i \geq 1$. This is precisely the reason why it is widely believed that for all $i \geq 1$, $\Sigma_i \neq \Pi_i$. This implies that $\mathsf{PH} \neq \Sigma_2$, a conjecture on which our result will be based on.

Probabilistic complexity classes

Randomized complexity classes are defined with respect to Turing machines which have access to an additional tape which contains an infinite number of uniform and independent random bits. For this paper, we are just concerned with *probabilistic polynomial time* Turing machines which always halt (independently of the random tape) after a polynomial number of steps (in the length of the input). Naturally, for an input x to such a randomized machine M, one associates probabilities to the computation $M(x)$. The class RP is the class of all languages L, such that there is a probabilistic polynomial time Turing machine M_L, such that for all $x \in L$, $\Pr[M_L(x) \text{ accepts}] = 1$ and for all $x \notin L$, $\Pr[M_L(x) \text{ accepts}] \leq 1/2$. The probabilistic complexity classes important for this paper will be RP and co-RP. Finally, we define the class $\mathsf{NP}^{\mathsf{co-RP}}$ as the collections of languages L, for which there is a probabilistic polynomial time machine M_L, and a polynomial $p(\cdot)$, such that if $x \in L$ there is a $y \in \{0,1\}^*$, with $|y| = O(p(|x|))$, $\Pr[M_L(x,y) \text{ accepts}] \leq 1/2$, and if $x \notin L$, then for all $y \in \{0,1\}^*$, with $|y| = O(p(|x|))$, $\Pr[M_L(x,y) \text{ accepts}] = 1$.

CHAPTER IV

LEARNING SYMMETRIC JUNTAS

4.1 Introduction

Motivation

An important problem in computational learning theory is to learn efficiently in the presence of irrelevant data. The following formalization is due to Blum [15] and Blum and Langley [18]:

> Let f be an unknown boolean function over an n bit domain which depends on only k bits. Assume k is much lesser than n, typically $k = O(\log n)$. Such a function is called *k-junta*. Given an oracle access to independent sample points $\langle \mathbf{x}, f(\mathbf{x}) \rangle$, where $\mathbf{x}$ is a uniformly and randomly chosen n bit string, what is the complexity of learning f?

More formally, a boolean function $f : \{0,1\}^n \mapsto \{0,1\}$ is said to *depend* on variable i, if there are vectors $\mathbf{x}$ and $\mathbf{y}$ that differ only in i'th position and $f(\mathbf{x}) \neq f(\mathbf{y})$. The variables on which f depends are called the *relevant* variables of f. A function that depends only on an (unknown) subset of k ($\ll n$) variables is called a *k-junta*. The class *k-juntas* is the set of all boolean functions on n variables which depend on at most k variables. Typically, k is of the order $\log n$. Hence, a running time that is polynomial in n is considered efficient.

It is believed [16, 88] that learning the class k-juntas is one of the most important open problem in the theory of uniform distribution learning. It has connections with learning DNF formulas and decision trees of super-constant size, see [25, 53, 83, 119, 120] for more details. The hardness of learning k-juntas has also been a basis of a cryptosystem in [17]. This gives another reason to investigate the complexity of this problem.

The first non-trivial result in learning juntas was obtained by Mossel, O'Donnell and Servedio in [88], who gave an algorithm that takes time roughly $n^{0.704k}$ to learn k-juntas.

They prove their result by analyzing the Fourier and $\mathbb{F}_2$ representations of a boolean function simultaneously and showing an elegant dichotomy theorem concerning these representation, see [88] for details.

Fourier based learning has enjoyed a tremendous success in the computational learning theory community. The primary reasons for this are two-fold:

- Fourier coefficients of boolean functions contain all the information about f.
- Fourier coefficients are easy to compute in the standard (PAC) learning theory setting.

Roughly, if one can show that every boolean function in a family has some Fourier coefficient of order at most t which is not zero, then one can learn this family[1] in time roughly n^t. Letting $t = k$, one can obtain a trivial n^k time algorithm for learning k-juntas. However, it is conjectured in [88] that, if a k-junta f has zero Fourier coefficients for all sets of size up to $2k/3$, then there is way to fix some $2k/3$ bits of it, such that, f restricted to this bit fixing, is a parity function (or its complement) in the remaining bits.[2] There are examples for which this conjecture, if true, would be the best possible. For instance, let $f(x_1, \ldots, x_n) := \text{MAJORITY}(x_1, \ldots, x_{2k/3}) \bigoplus (x_{2k/3+1} \oplus \cdots \oplus x_k)$.[3] However, there is a simple way to learn this function in time $n^{k/3}$ [16]. Blum, in [16], raises as an open to decide if one can come up with a better learning algorithm even for this special class of functions! It seems that substantially new ideas would to be needed to go beyond this $n^{k/3}$ barrier. Most likely, these will have consequences beyond learning juntas. In the dearth of such approaches, a natural question is:

> What are the sub-families of k-juntas for which Fourier based techniques will lead to fast learning algorithms?

Another important thing to notice about the function

$$\text{MAJORITY}(x_1, \ldots, x_{2k/3}) \bigoplus (x_{2k/3+1} \oplus \cdots \oplus x_k)$$

[1] It should also be possible to compute the Fourier coefficients of order upto t efficiently.
[2] The proof of this would imply a learning algorithm for the class of k-juntas in time $n^{2k/3}$.
[3] For the ease of notation, we let $x_1, \ldots, x_k$ be some *unknown* variables from among $x_1, \ldots, x_n$.

is that there is a huge sub-group of the symmetric group (S_k) acting on it, namely $S_{2k/3} \times S_{k/3}$. In light of these considerations, the most natural, and arguably an important sub-family, that comes up is the class of symmetric k-juntas. This is the class of all k-juntas which are symmetric in the variables they depend on. Surprisingly, even for this family, the best known algorithm, before this work, ran in time $n^{2k/3}$ [88]. They consider two cases:

- Unbalanced case: $\Pr[f(\mathbf{x}) = 0] \neq \Pr[f(\mathbf{x}) = 1]$.
- Balanced case: $\Pr[f(\mathbf{x}) = 0] = \Pr[f(\mathbf{x}) = 1]$.

In the first case, it is shown, using elementary arguments, that f has a non-zero Fourier coefficient of order at most $2k/3$. While in the second case, they invoke a result due to von zur Gathen and Roche [123], which implies that Fourier coefficients of order $o(k)$ suffice. Hence, combining these two cases, they obtain that $t = 2k/3$ is sufficient. Neither of these results is sufficient to go below the $2k/3$ barrier. Though the result of von zur Gathen and Roche is quite strong, there does not seem to be a way to modify it to handle the unbalanced case. Further, the techniques of [88] do not seem to be powerful enough to improve on $2k/3$. The fact that the problem at hand turns out to be equivalent to a question about a system of Diophantine equations, calls for number theoretic techniques, and it is unlikely that one will be able to make much progress just by elementary arguments. We should also remark here that the techniques of [123] bear some resemblance to ours, but only at a superficial level.

This is the motivation for us to study the following structural result about the Fourier representation of symmetric boolean functions:

> What is the smallest t such that for every symmetric boolean function on k variables, there is a Fourier coefficient of order at most t which is not zero ?

Notice that if f is the parity function, or its complement, then $t = k$. Hence, this question is interesting only when one prohibits f to be one of these two functions. Let $\tau(k)$ be the smallest such t in this case. In this chapter, we study this extremal parameter.

The main contribution of this chapter is a proof of the following *self similar* nature of this question:

$$\text{If } \tau(l) \leq s, \text{ then for } k \geq k_0(l),\ \tau(k) \leq \tfrac{s+1}{l+1}k.$$

Coupled with preliminary computer based calculations for $\tau(l)$ for small values of l, we obtain that $\tau(k) \leq 3k/19$. We believe that these results are bound to improve with more sophisticated calculations for $\tau(l)$ for small values of l. As of now, these imply a learning algorithm for the class of symmetric k-juntas in time about $n^{3k/19}$. We hope that our line of work will finally result in an efficient learning algorithm for symmetric juntas.

Technically, studying this extremal parameter is equivalent to the study of a system of Diophantine equations. As a first step, we simplify these Diophantine equations. This is done by moving to a representation which is equivalent to the Fourier representation, but seems much simpler for the application of number theoretic tools. Once this is done, we study these Diophantine equations over local fields and combine the solutions in a combinatorial manner. The following self-similarity of Pascal's Triangle plays an important role. Recall that the m-th row of Pascal's Triangle consists of binomial coefficients of the form $\binom{m}{j}$, for $0 \leq j \leq m$. If $m = lp$ for some l, and some prime p, then the values obtained by reducing this row modulo p, can be read off directly from the l-th row of Pascal's Triangle!

4.2 Our contribution

Results

The main result of this chapter concerns the structure of Fourier coefficients of symmetric boolean functions. The following is a statement of the main theorem [4].

Theorem 4.2.1. *Let $0 < s \leq l$ be integers, $\epsilon > 0$, and f a symmetric boolean function on k variables other than parity or its complement. Further, assume that for all symmetric boolean functions on l-variables other than parity or its complement, there is a Fourier*

[4]The hypothesis in this statement is slightly different (though equivalent) to the theorem we will actually prove, and requires introduction of substantial notation. Hence, we defer the formal statement till Section 4.5.

coefficient of order at most s which is non-zero. Then, there exists a constant $k_0 := k_0(\epsilon, s, l)$ such that, for all $k \geq k_0$, there is an integer $0 < t \leq \left(\frac{s+1}{l+1} + \epsilon\right) k$, such that f has a non-zero Fourier coefficient of order t.

As will be clear later, due to the nature of the function $k_0(\epsilon, s, l)$, we choose l to be a constant. To obtain siginificant results, we performed a (brute force) computer search to find the l (and s) which minimize $\frac{s+1}{l+1}$. It turns out, that $l = 18$ and $s = 2$ is the instance that minimizes $\frac{s+1}{l+1}$, among the ones we ran a search for. These yield the following result about learning symmetric juntas.

Theorem 4.2.2. *The class of symmetric k-juntas can be learned exactly under the uniform distribution with confidence $1 - \delta$ in time $n^{\frac{3k}{19}+o(k)} \cdot \mathrm{poly}(2^k, n, \log(1/\delta))$.*

A remark on the computational search

We remark that $3k/19$ in the statement of Theorem 4.2.2 and can probably be driven down by a more powerful and sophisticated search for the finite case. The emphasis of this presentation is not on this computational aspect, and we hope to expand on this in the final version of the chapter related to this work. We conjecture that all symmetric boolean functions on k variables, other than parity or its complement, have a non-zero Fourier coefficient of order at most 4, proving this at this point seems quite hard. There is some hope though, that we can prove that there are infinitely many k for which this conjecture is true. Combining such a result with Theorem 4.2.1, one would immediately obtain a learning algorithm which runs in time $n^{\gamma k} \cdot \mathrm{poly}(2^k, n, \log(1/\delta))$, for all $\gamma > 0$.

Significance of the results

- We provide a *novel* framework to understand the Fourier coefficients of boolean functions, and in particular, for symmetric functions. Though there is an explicit formula for Fourier coefficients of symmetric boolean functions in terms of the so called *Krawtchouk Polynomials* [13], these expressions are not so amenable. Our framework, though equivalent, is more susceptible to number-theoretic techniques. This may be

one reasons why these result were not obtained earlier and we hope that this will find future use in studying symmetric boolean functions.

- Theorem 4.2.1 provides insight into the structure of Fourier coefficients of symmetric boolean functions. A particularly appealing feature of the result is that one can obtain quantitatively better versions of Theorem 4.2.2 by proving (or more precisely verifying) a problem of constant size. Indeed, it turns out that this *self similarity* is really a consequence of the self-similar or fractal nature of the Pascal's Triangle of binomial coefficients [45].

- In lieu of the fact that one needs fundamentally new learning techniques (beyond the usual Fourier based approach) to learn the class of k-juntas in time better than $O(n^{k/3})$ (see the introduction), theoretically, it becomes important to understand what is the largest sub-class of k-juntas that can be learnt much faster than this $n^{k/3}$ barrier. Symmetric k-juntas seem to be a natural and rich sub-class of k-juntas, for which prior to our work, the best known algorithm ran in time $n^{2k/3}$ [88].

  It seems that our techniques can possibly be extended to obtain an efficient learning algorithm for symmetric juntas. But this would require proving some hard number-theoretic results, which are elusive at this moment.

Before presenting an overview of the proof techniques, it is useful at this point, to review some fundamental facts about boolean functions.

4.2.1 Fourier coefficients of boolean functions

We consider boolean functions from $\{1,-1\}^k \mapsto \{1,-1\}$ and $\{0,1\}^k \mapsto \{0,1\}$. (As is usual, 0 is identified with 1 and 1 with -1. By abuse of notation we will consider f as a boolean function in both $\{0,1\}$ and $\{1,-1\}$ notations. We refer to f as a boolean function on k variables.) For a set $S \subseteq [k]$, define $\chi_S : \{1,-1\}^k \mapsto \{1,-1\}$ to be the function $\chi_S(\mathbf{x}) := \prod_{i \in S} x_i$. (By convention, the boldface $\mathbf{x}$ denotes the vector $(x_1, \ldots, x_k)$.) For a function $f : \{1,-1\}^k \mapsto \{1,-1\}$, $S \subseteq [k]$, define the *Fourier coefficient* corresponding to S as $\hat{f}(S) := \frac{1}{2^k} \sum_{\mathbf{x} \in \{1,-1\}^k} f(\mathbf{x}) \chi_S(\mathbf{x})$. The Fourier expansion of f is: $f(\mathbf{x}) = \sum_{S \subseteq [k]} \hat{f}(S) \chi_S(\mathbf{x})$.

If f is a symmetric, f is completely determined by its value on any $k+1$ vectors of distinct weights[5], we use the following vector representation of f: $\nu(f) := (f_0, f_1, \ldots, f_k)^T$. Here f_i is the value of f on a vector of weight i. Furter f has precisely $k+1$ (non-equivalent) Fourier coefficients, $(\hat{f}_0, \ldots, \hat{f}_k)$. Here $\hat{f}_t$ is defined as $\hat{f}(S)$, for some $S \subseteq [k]$ with cardinality t. Since f is symmetric, this does not depend on the choice of S. There is an explicit formula for $\hat{f}_t$ in terms of the *Krawtchouk Polynomials* [13], but these formulas are not very amenable. The following four special symmetric functions on k variables will appear often, and we wish to denote them by: the two constant functions $\mathbf{0}$ and $\mathbf{1}$, and the parity function $\oplus$, and its complement $\overline{\oplus}$.

Technical overview

The first step in the proof is to look at a representation that is *equivalent* to the Fourier representation, but turns out to be much simpler for the purpose of our analysis. Let f be a symmetric boolean functions on k variables. Assume for the moment that f is not constant or parity or its complement.

For a set $S \subseteq [k]$ and a string $\sigma \in \{0,1\}^t$, define $p_{S,\sigma}(f) := \Pr[\mathbf{x}_S = \sigma | f(\mathbf{x}) = 1]$. (For $\mathbf{x} = (x_1, \ldots, x_k)$, $\mathbf{x}_S$ denotes the ordered $|S|$-tuple of x_i-s, for $i \in S$. Also, we refer to $p_{S,\sigma}$ as S, σ-projections.) If f is symmetric, this quantity just depends on the size of the set, say s, and the weight of σ, say w. In this case, we abbreviate $p_{S,\sigma}$ by $p_{s,w}$. Call a function f t-null, if for all sets of size t and all $0 \leq w \leq t$, $p_{t,w}(f) = 1/2^t$. We show that if for some t, $\hat{f}_t \neq 0$, then f is not t-null. The converse also holds.

In turn, the t-nullity conditions for f suggest that f is a 0/1 solution to the following system of Diophantine equations:

$$A_{k,t}\mathbf{x} = c\mathbf{1}.$$

Here $A_{k,t}$ is a $t+1 \times k+1$ matrix, whose rows are indexed by $\{0, 1, \ldots, t\}$, columns are indexed by $\{0, 1, \ldots, k\}$, and the i, j-th entry is $\binom{k-t}{j-i}$ (This means that there is a constant c, such that the vector $\nu(f)$ is a solution to this system.)

[5]The *weight* of a boolean vector is defined to be the number of 1s in it.

Thus, the problem is reduced to a completely number theoretic one. It is easy to see the when $c = 0$, the all zeros vector satisfies this system. Similarly for $c = 2^{k-t}$ we get the all ones vector as a solution. Among other trivial solutions are for 2^{k-t-1}, the parity function and its complement.

It is clear that increasing the parameter t puts more and more constraints on this system. The question now becomes what is the smallest number of such equations needed, so that the only solutions are the trivial ones (mentioned above)?

First, we show that reducing this system of equations modulo a carefully chosen prime, reduces to the same problem, albeit, of a smaller size. In fact, we choose the prime so that reducing these equations modulo it, we get (many disjoint) constant sized instances of a *similar* set of Diophantine equations. This is the *self similarity* of these Diophantine equations we exploit. (It turns out, that this fact is essentially a consequence of the self similar behavior of the rows of Pascal's Triangle.)

Next, we invoke results for constant-sized Diophantine equations (which in our case are gotten by a computer search). Since there is some freedom to the solution space for each such constant-sized system, we are still left with exponentially many choices for f.

We handle this last problem by combining the results in a clever way (by yet another choice of a prime) to obtain that f could be only one of the four functions: constant, parity or its complement. This is a combinatorial argument.

Organization

For the ease of understanding the results, first, we present a self-contained exposition of a special case in Section 4.4. In Section 4.5, we restate and prove Theorem 4.2.1 in its full generality. In Section 4.3, we give the mathematical tools to be used in the proof of these results. Finally, in Section 4.7 we present a proof of Theorem 4.2.2. This section also includes some preliminaries from learning theory. In Section 4.9, we give results of our preliminary computational experiments.

4.3 Main tools

In this section we present the tools used in proving our results.

4.3.1 An arithmetic property of binomial coefficients

The next easy result is a special case of Lucas' Theorem [45]. This result illustrates the *self similar* nature of the Pascal's Triangle modulo primes.

Lemma 4.3.1. *For a prime p, any $l \geq 0$ and $0 \leq i \leq lp$, $\binom{lp}{i} \equiv \binom{l}{j} \mod p$ if $i = jp$ for some $0 \leq j \leq l$, and 0 otherwise.*

Proof. For a prime p, and $0 \leq j \leq p$, $\binom{p}{j} \equiv 1 \mod p$ if $j = 0$ or $j = p$, and $\binom{p}{j} \equiv 0 \mod p$ otherwise. Hence, for an indeterminate x, $(1+x)^p \equiv 1 + x^p \mod p$. Consider $\sum_{i=0}^{lp} \binom{lp}{i} x^i = (1+x)^{lp} = ((1+x)^p)^l$. Reducing this sum modulo p, and using the fact above, one obtains

$$\sum_{i=0}^{lp} \binom{lp}{i} x^i \equiv (1+x^p)^l \mod p.$$

But $(1+x^p)^l = \sum_{j=0}^{l} \binom{l}{j} x^{pj}$. Comparing coefficients of x^{pj} on both sides of the above equation, one gets the desired conclusion. □

On numerous occasions, we will use the following result about the density of primes. This is an easy corollary of the celebrated Prime Number Theorem.

Lemma 4.3.2. *For large enough n, there is a prime $p \leq n$, such that $p = n - o(n)$.*

4.3.2 S, σ-projections of boolean functions

In this section, we introduce a *novel* way of looking at Fourier coefficients, in terms of certain probabilities, which we describe next.

Let $f : \{0,1\}^k \mapsto \{0,1\}$ be a boolean function, with $\Pr[f(\mathbf{x}) = 1] \geq \Pr[f(\mathbf{x}) = 0]$. [6] Unless mentioned, in this section, all probabilities are over the uniform distribution. Recall, that for a vector $\mathbf{x} = (x_1, \ldots, x_k)$, and a set $S \subseteq [k]$, $\mathbf{x}_S$ is the projection of $\mathbf{x}$ on the indices of S. Let $\sigma \in \{0,1\}^{|S|}$. Define the *S, σ-projection* of f to be:

$$p_{S,\sigma}(f) := \Pr\left[\mathbf{x}_S = \sigma | f(\mathbf{x}) = 1\right].$$

[6] For our learning theory application, this is sufficient. To be mathematically precise, one should also handle the symmetric case $\Pr[f(\mathbf{x}) = 0] \geq \Pr[f(\mathbf{x}) = 1]$.

Call a boolean function f on k variables *t-null*, if for all sets $S \subseteq [k]$, with $|S| = t$, and for all $\sigma \in \{0,1\}^t$, $p_{S,\sigma}(f) = \frac{1}{2^t}$. The following lemma connects the S,σ-projections of f to its Fourier coefficients.

Lemma 4.3.3. *Let f be a boolean function on k variables. f is t-null for some $1 \leq t \leq k$, if and only if, for all $\emptyset \neq S \subseteq [k]$ with cardinality at most t, $\hat{f}(S) = 0$.*

Proof. For some $\sigma \in \{0,1\}^t$, and $T \subseteq [k], |T| = t$, let g be the function obtained by substituting σ for $\mathbf{x}$ in f. Even though g is a function of $\mathbf{x}_{T^c}$, we write $g(\mathbf{x})$ for ease of notation. By chain rule for conditional probabilities,

$$p_{T,\sigma}(f) = \frac{\Pr[f(\mathbf{x}) = 1 | \mathbf{x}_T = \sigma] \cdot \Pr[\mathbf{x}_T = \sigma]}{\Pr[f(\mathbf{x}) = 1]} = \frac{\Pr[g(\mathbf{x}) = 1] \cdot 2^{-t}}{\Pr[f(\mathbf{x}) = 1]}$$

Hence, $p_{T,\sigma}(f) = 2^{-t}$ if and only if $\Pr[g(\mathbf{x}) = 1] = \Pr[f(\mathbf{x}) = 1]$, or equivalently, $\hat{f}(\emptyset) = \hat{g}(\emptyset)$. This is true if and only if $\hat{f}(S) = 0$ for all $\emptyset \neq S \subseteq T$. The lemma follows. □

The following is an immediate corollary of this lemma.

Corollary 4.3.4. *Let f be a boolean function on k variables. If f is t-null for some $1 \leq t \leq n$ then f is s-null for $1 \leq s \leq t$.*

Moreover, for symmetric functions, S,σ-projections of f just depend on $s := |S|$ and the weight w of σ. (Here weight of a vector $\mathrm{wt}(\sigma)$ is the number of 1-s in σ.) Hence, it is sufficient to consider s,w-projections of f. We denote them by $p_{s,w}(f)$. For a symmetric function f, $\Pr[f(\mathbf{x}) = 1] = 2^{-k} \sum_{i=0}^{k} \binom{k}{i} f_i$. Further, if $\mathbf{x}_s$ denotes $\mathbf{x}_{\{1,\ldots,s\}}$ and σ is a string of weight w, it is easy to see that

$$\Pr[f(\mathbf{x}) = 1 \wedge \mathbf{x}_s = \sigma] = 2^{-k} \sum_{i=0}^{k} \binom{k-s}{i-w} f_i.$$

If f is s-null, then for any $0 \leq w \leq s$, $\Pr[f(\mathbf{x}) = 1 \wedge \mathbf{x}_s = \sigma] = p_{s,w}(f) \cdot \Pr[f(\mathbf{x}) = 1] = 2^{-(s+k)} \cdot \sum_{i=0}^{k} \binom{k}{i} f_i$, which is independent of w. Hence, there exists a constant $c := c(f,s,k)$ such that

$$\sum_{i=0}^{k} \binom{k-s}{i-w} f_i = c, \quad \forall\ 0 \leq w \leq s. \tag{5}$$

It is easy to see that boolean functions $\{\mathbf{0}, \mathbf{1}, \oplus, \overline{\oplus}\}$ satisfy this system of equations for all s, albeit, for different constants. The question is how large must s be so that these 4 boolean functions are the only solutions to this system, for any constant on the right hand size? To summarize:

Lemma 4.3.5. *For $1 \leq s \leq k$, let $A_{k,s}$ be the $(s+1) \times (k+1)$ matrix*

$$A_{k,s}(i,j) := \binom{k-s}{j-i}.$$

If f is s-null, then there is a positive integer $c := c(f,s,k)$ such that[7]

$$A_{k,s}\nu(f) = c\mathbf{1}.$$

4.4 The case of $\frac{\mathbf{k}}{\mathbf{2}}$

In this section we give a self-contained proof of the following (weaker) result. The aim is to illustrate the key ideas behind the proof of Theorem 4.2.1, without worrying about all the tedious details. (Note that this result already leads to an improvement over the result of [88] on learning symmetric juntas!)

Theorem 4.4.1. *For any symmetric boolean function f on k variables, which is not one of $\{\mathbf{0}, \mathbf{1}, \oplus, \overline{\oplus}\}$, there is a $t = \frac{k}{2} + o(k)$ such that $\hat{f}_t \neq 0$.*

The following lemma combined with Lemma 4.3.2 immediately implies Theorem 4.4.1.

Lemma 4.4.2. *Let k be large enough and $2 < p < q \leq k/2$ be two primes. Let f be a non-constant symmetric boolean function on k variables. If f is $(k-p)$-null (and hence, $(k-q)$-null, then $f \in \{\oplus, \bar{\oplus}\}$.*

(In what follows we assume $k/2$ is an integer to make the presentation clean. One should replace $k/2$ by $\lfloor k/2 \rfloor$ to be precise.)

To prove this theorem, we employ a sieving technique to analyze Equation (5). The idea is to consider $A_{k,s,p} := A_{k,s} \mod p$ for a prime p. In particular, when $k - s = p$, we get from Lemma 4.3.1 that the equations are of the form $f_i + f_{i+p} = c_1$ for $0 \leq i \leq k-p$,

[7] *Here $\binom{l}{m}$ is 0 if $m < 0$ or $m > l$ and $\mathbf{1}$ denotes the all 1-s vector of dimension $k+1$.*

where $c_1 \equiv c(f,s,k) \mod p$. Since the f_i's are either 0 or 1, c_1 is either $0, 1$ or 2. In fact, if $c_1 = 0$, then $f_i = 0$ for all i, and if $c_1 = 2$, then $f_i = 1$ for all i. (Assume $p > 2$.)

We need the following combinatorial lemma. For positive integers $k, p \neq q$, let $G_{k,p,q}$ be the graph with vertex set $\{0, 1, 2, \ldots, k-1, k\}$, and $0 \leq i \neq j \leq k$ are adjacent whenever $|i - j| = p$ or q.

Lemma 4.4.3. *For positive integers k, p, q such that $(p, q) = 1$ and $p + q \leq k$, $G_{k,p,q}$ is connected.*

Proof. We proceed by induction on $\min\{p, q\}$. Without loss of generality, let $p > q$. Clearly, the lemma holds for the base case, $q = 1$. Let $0 \leq i < j \leq k$, and $j - i = p - q$. Since $p + q \leq k$, either $i + p \leq k$ or $i - q \geq 0$. In either case, there is a path between i and j. Hence, it is enough to show that $G_{k,p-q,q}$ is connected, which follows by the induction hypothesis. □

Proof of Lemma 4.4.2. We show that the system of equations (5), put enough restrictions on $\nu(f)$, so that it can correspond to at most 2 non-constant boolean symmetric functions. Since the functions $\{\oplus, \overline{\oplus}\}$ satisfy (5), f has to be one of these!

Since f is $(k-p)$-null, and $k - p > k - q$, by Lemma 4.3.4, f is $(k-q)$-null. Hence, by Lemma 4.3.5, $\exists c_1, c_2$ such that

$$A_{k,k-p}\nu(f) = c_1\mathbf{1} \quad \text{and} \quad A_{k,k-q}\nu(f) = c_2\mathbf{1}.$$

Consider these two system of equations modulo p and q respectively. Let $c_p \equiv c_1 \mod p$, and $c_q \equiv c_2 \mod q$. They become (respectively in $\mathbb{F}_p$ and $\mathbb{F}_q$)

$$A_{k,k-p,p}\nu(f) = c_p\mathbf{1} \quad \text{and} \quad A_{k,k-q,q}\nu(f) = c_q\mathbf{1}.$$

Moreover, f being non-constant and $p, q > 2$, $c_p = c_q = 1$. Therefore, the equations are of the form $f_i + f_j = 1$ for all $0 \leq i \neq j \leq k, |i - j| = p$ or q. Since $G_{k,p,q}$ is connected (Lemma 4.4.3) it follows that fixing the value of any one f_i uniquely determines f, and hence, there are at most 2 possible choices for f, namely, $\{\oplus, \overline{\oplus}\}$. □

4.5 The general case

The hypothesis for the finite case and Theorem 4.2.1 restated

In this section we give a formal version of Theorem 4.2.1. First we need the following hypothesis, which we refer to as $\mathcal{H}(u, m)$.

Hypothesis 4.5.1. *For integers $0 < u \leq m$, we say that $\mathcal{H}(u, m)$ is true if for all $\mathbf{x} \in \{0,1\}^{m+1}$, and any integer $c \geq 0$, there are at most four solutions (from the set $\{\mathbf{0}, \mathbf{1}, \oplus, \overline{\oplus}\}$) to the following system of equations:*

$$A_{m,u}\mathbf{x} = c.$$

As usual, $\nu(\mathbf{0}) := (0, \ldots, 0)$, $\nu(\mathbf{1}) := (1, \ldots, 1)$, $\nu(\oplus) := (0, 1, \ldots, 1/2(1 + (-1)^{m+1}))$, and $\nu(\overline{\oplus}) := (1, 0, \ldots, 1/2(1 - (-1)^{m+1})$. We say that $\mathcal{H}(m, u)$ is false if it is not true.

In this language, the version of Theorem 4.2.1 we prove is the following. It is immediate that $\mathcal{H}(l + s, s)$ implies the hypothesis of Theorem 4.2.1. Moreover, it is not too difficult to see that in fact, the two hypotheses are equivalent. We omit the details. It should be noted that our computer search tests $\mathcal{H}(m, u)$.

Theorem 4.5.2. *Let $0 < s \leq l$ be integers, $\epsilon > 0$, and f a symmetric boolean function on k variables other than $\{\mathbf{0}, \mathbf{1}, \oplus, \overline{\oplus}\}$. Further, assume that $\mathcal{H}(l+s, s)$ is true. Then, there exists a constant $k_0 := k_0(\epsilon, s, l)$ such that, for all $k \geq k_0$, there is an integer $0 < t \leq \left(\frac{s+1}{l+s+1} + \epsilon\right) k$, such that f is not t-null.*

Proof of the main theorem

In this section we present a sketch of proof Theorem 4.5.2. First, assume that there is a prime p such that $k = (l + s + 1)p - 1$. Let $t = k - lp = (s + 1)p - 1$. We handle the case when this is not the case later.

Assume on the contrary that $\nu(f) = (f_0, f_1, \ldots, f_k)$ is t-null. By definition there is a constant c such that the following is true.

$$A_{k,t}\nu(f) = c. \tag{6}$$

Reducing to a finite problem

Notice that by definition, $k-t = lp$. For $0 \leq i \leq p-1$, let $\mathbf{F}_i := (f_i, f_{i+p}, f_{i+2p}, \ldots, f_{i+(l+s)p})$. Hence, reducing Equations 6 modulo p, and using Lemma 4.3.1, one obtains the following systems of equations.

$$\begin{aligned} A_{l+s,s}\mathbf{F}_0 &\equiv c' \mod p \\ A_{l+s,s}\mathbf{F}_1 &\equiv c' \mod p \\ &\vdots \\ A_{l+s,s}\mathbf{F}_{p-1} &\equiv c' \mod p \end{aligned}$$

Here $c' \equiv c \mod p$. As in the proof of Theorem 4.4.1, choosing $p > 2^{l+s}$, one sees that these modular equations are in fact exact. That is, there is a positive integer $d \geq 0$, such that the following hold.

$$\begin{aligned} A_{l+s,s}\mathbf{F}_0 &\equiv d \\ A_{l+s,s}\mathbf{F}_1 &\equiv d \\ &\vdots \\ A_{l+s,s}\mathbf{F}_{p-1} &\equiv d \end{aligned} \tag{7}$$

Using the fact that $\mathcal{H}(l+s, s)$ is true, one applies them to $\mathbf{F}_0, \ldots, \mathbf{F}_{p-1}$ to obtain that there are at most 4^p choices for f. To narrow these down to 4, we need to be slightly more clever.

Combining the finite instances

Let $2 \leq k/2 < q \leq (l+1)p$ be a prime. Since f is t-null, by Corollary 4.3.4, f is $k-q$-null. Now, consider system of equations $A_{k,k-q}\nu(f) = c$ modulo the prime q. Since $q > 2$, we get, for some $e \geq 0$, exact equations of the following form:

$$\begin{aligned} f_0 + f_q &= e \\ f_1 + f_{q+1} &= e \\ &\vdots \\ f_{k-q} + f_q &= e. \end{aligned} \tag{8}$$

The idea is that these equations combined with Equations 7 are sufficient to restrict f to one of the four functions, as desired. This is what we proceed to do. First we need a technical lemma. For an integer $m \geq 0$, let $(m)_p := m \mod p$. Also, for $0 \leq i \leq p-1$, let $[iq]_p := \{(iq)_p, (iq)_p + p, \ldots, (iq)_p + (l+s)p\}$. The following lemma is obvious.

Lemma 4.5.3. *For $0 \leq i < j \leq p-1$, and primes p and q, $[iq]_p \cap [jq]_p = \emptyset$.*

Now, fix f_0 and f_p. This fixes all the indices in $\mathbf{F}_0$, as $\mathcal{H}(l+s, s)$ is true for $\mathbf{F}_0$. Now, using Equations 8, we get that f_q and f_{q+p} are fixed. Using Equations 7, this implies that all the indices in $\mathbf{F}_{(q)_p}$ get fixed. Iterating the alternate use of these two system of equations, along with Lemma 4.5.3, one obtains that all of f is determined once f_0 and f_p are fixed. Hence, this f has at most four choices: $\{\mathbf{0}, \mathbf{1}, \oplus, \overline{\oplus}\}$.

Thus, for $k = (l+s+1)p - 1$ and $t = (s+1)p - 1$, given $\epsilon > 0$, we can choose k_0, such that for $k \geq k_0$, $t \leq \left(\frac{s+1}{l+s+1} + \epsilon\right) k$, and hence, a contradiction is reached.

Handling the residual class of variables

The only case remains when there is no prime p such that $k = (l+s+1)p - 1$. In this case, we pick a prime p in the interval $\left[\frac{k}{l+s+1} - o(k), \frac{k}{l+s+1}\right]$. We are guaranteed the existence of such a prime by Lemma 4.3.2. Let $t = k - lp$. Hence, $(s+1)p + o(p) \geq t \geq (s+1)p$. Since we think of l as a constant, $p = \Omega(k)$. Hence, there is a small number of variables ($o(k)$) which remain to be dealt with in the previous argument. But one can verify that Equations 8 are sufficient to fix this very small class of residual variables too! Hence, in this case too, we reach a contradiction. This completes the proof of the theorem.

4.6 The finite case

Using a computer, we verified (among others) that $\mathcal{H}(18,2)$ is true. Hence, plugging this in Theorem 4.5.2, we obtain the following corollary.

Corollary 4.6.1. *Let f be a symmetric boolean function on k variables other than $\{\oplus, \overline{\oplus}\}$. Then, there exists a constant $k_0 := k_0(\epsilon)$ such that, for all $k \geq k_0$, there is an integer $0 < t \leq \left(\frac{3}{19} + \epsilon\right) k$, such that f has a non-zero Fourier coefficient of order t.*

We must mention that we can prove that $\mathcal{H}(18,2)$ is true, without resorting to a computational proof. The proof however, is quite long and nothing more than a tedious case analysis. The details will be included in the final version of this work.

4.7 Learning symmetric juntas

In this section we apply the results proved earlier (in particular Corollary 4.6.1) to obtain fast learning algorithms for the class of symmetric k-juntas on n variables. First we need some preliminaries and well known tools from computational learning theory.

4.7.1 The PAC learning model

Valiant [116] considered the following computational model of learning, which he called *Probably Approximately Correct* or PAC. The learning problem at hand is a *Concept Class* $\mathcal{C} = \cup_n \mathcal{C}_n$, where each $\mathcal{C}_n$ is a collection of boolean functions from $\{0,1\}^n \mapsto \{0,1\}$. Let ϵ be an *accuracy parameter* and δ a *confidence parameter*. A learning algorithm $\mathcal{A}$ for $\mathcal{C}$, takes ϵ, δ as its input, has access to an *oracle* $\mathcal{I}(f)$. For $f \in \mathcal{C}_n$, a query to $\mathcal{I}(f)$ outputs an *instance* $\langle \mathbf{x}, f(\mathbf{x}) \rangle$, where $\mathbf{x} \in_r \{0,1\}^n$. $\mathcal{A}$ is supposed to output a *hypothesis* $h : \{0,1\}^n \mapsto \{0,1\}$, such that $\Pr_{\mathbf{x} \in_r \{0,1\}^n}[h(\mathbf{x}) = f(\mathbf{x})] \geq 1 - \epsilon$. $\mathcal{A}$ is said to be a learning algorithm for class $\mathcal{C}$ if for all $f \in \mathcal{C}$, when $\mathcal{A}$ is run with oracle $\mathcal{I}(f)$, with probability at least $1 - \delta$ it outputs a hypothesis h such that $\Pr_{\mathbf{x}}[h(\mathbf{x}) = f(\mathbf{x})] \geq 1 - \epsilon$.

Remark 4.7.1. *Although Valiant's PAC model is defined for general distributions, in this chapter we will just be concerned with uniform distribution.*

4.7.2 k-junta

We recall the definition of a k-junta. Let $f : \{0,1\}^n \mapsto \{0,1\}$ be a boolean function. We say that f *depends* on the variable i, if there are vectors $\mathbf{x}$ and $\mathbf{y}$ that differ only in i'th position and $f(\mathbf{x}) \neq f(\mathbf{y})$. A function that depends only on an (unknown) subset of $k \ll n$ variables is called a k-junta. The variables on which f depends are called the *relevant* variables of f. Typically $k = O(\log n)$. Hence, a running time that is polynomial in $2^k, n$ and $\log(1/\delta)$ is considered efficient.

A symmetric k-junta is a boolean function, which has the additional property that it is symmetric in the variables it depends on. The class of all such functions defined on n variables is the class of symmetric k-juntas. In this section, we present an algorithm for learning this class in the uniform PAC model.

First, we summarize the techniques from learning theory that will be useful. The following theorem says that it is enough to learn one relevant variable of f.

Theorem 4.7.2. *[88] Suppose that there is a PAC learning algorithm that for any non-constant k-junta f identifies at least 1 relevant variable of f with confidence $1 - \delta$ in time $n^\alpha \cdot \text{poly}(2^k, n, \log(1/\delta))$, then there is a PAC learning algorithm that exactly learns f in time $n^\alpha \cdot \text{poly}(2^k, n, \log(1/\delta))$.*

4.7.3 Fourier coefficients and computational learning theory

Fourier coefficients are very important tools in learning theory. The following theorem says that one can compute these efficiently in the (uniform) PAC learning model.

Theorem 4.7.3. *Given access to uniform random instances from $f : \{0,1\}^n \mapsto \{0,1\}$, for any set $S \subseteq [n]$, there is a randomized algorithm which computes $\hat{f}(S)$ to within an additive error λ with confidence $1 - \delta$ using $O(\lambda^{-2} \log(1/\delta))$ instances and time.*

If $S \subseteq [n], S \neq \emptyset$ is such that $\hat{f}(S) \neq 0$, then all the variables in S are relevant variables. A natural algorithm is to run over all choices of S until we find an S so that $\hat{f}(S)$ is non zero. However, this could potentially take as long as n^k since some functions have $\hat{f}(S) = 0$ for all $S \subseteq [n], 1 \leq |S| \leq k-1$. One such example is the parity function,

PARITY$(x_1, \ldots, x_n) := x_{i_1} \oplus \cdots \oplus x_{i_k}$. Fortunately, if we know that f is a parity function (or its complement), then it can be learnt quickly.

Theorem 4.7.4. *Let $f : \{0,1\}^n \mapsto \{0,1\}$ be the parity function or its complement on an unknown subset of the boolean variables $x_1, \ldots, x_n$. There is a PAC learning algorithm $\mathcal{A}$ (given access to a uniform oracle for f) which outputs a hypothesis h, such that with probability at least $1-\delta$, $\Pr[h(x) = f(x)] \geq 1-\epsilon$. Moreover $\mathcal{A}$ runs in time $O\left(\left(\frac{n}{\epsilon} + \frac{\log 1/\delta}{\epsilon}\right)^3\right)$.*

The proof of both these theorems are easy and can be found for instance in [88].

Now, consider the following algorithm to learn a sub-family $\mathcal{F}$ of k-juntas:

> Given $f \in \mathcal{F}$ in the uniform PAC setting, compute $\hat{f}(S)$ for all $S \subseteq [n], 1 \leq |S| \leq t$. If for all such S, $\hat{f}(S) = 0$, run the algorithm to learn if f is a parity or its complement of some subset of $[n]$.

Of course, if we let $t = k$, this algorithm works for any sub-family $\mathcal{F}$ of the class k-juntas. Moreover, for the class k-juntas, this algorithm is not known to any better than the trivial bound of n^k. It is conjectured in [88], that, if a k-junta f has zero Fourier coefficients for all sets of size up to $2k/3$, then there is way to fix some $2k/3$ bits of it, such that, f restricted to this bit fixing, is a parity function (or its complement) in the remaining bits. A proof of this conjecture would result in a learning algorithm for k-juntas that runs in time $n^{2k/3}$.

For symmetric k-juntas, an algorithm that runs in time $n^{2k/3}$ is given in [88]. This is done by showing that one can choose $t = 2k/3$ for the above learning algorithm. If we are guaranteed that the only functions that have $\hat{f}(S) = 0$ for all $S \subseteq [n], 1 \leq |S| \leq t$ are parity and its complement, then this algorithm learns f in time $n^t \cdot \text{poly}(2^k, n, \log(1/\delta))$. This is exactly the content of our work, summarized in Corollary 4.6.1. Thus, in this setting, one immediately derives Theorem 4.2.2 as a direct consequence of Corollary 4.6.1.

4.8 Conclusion

The most important open problem that remains is to ascertain the true behavior of the function $\tau(k)$. There is some computational evidence that $\tau(k) \leq 4$, see Section 4.9. Resolving

this seems hard at this moment. A relatively easier problem, which seems approachable, is to show that $\tau(k)$ is at most a constant for infinitely many k. As noticed earlier, this will already imply $\tau(k) \leq \epsilon k$, for all $\epsilon > 0$, for large enough k. Among other problems, it will be very useful to be able to determine quickly if $\mathcal{H}(l, s)$ is true or not. Right now, we know of no method other than essentially an exponential algorithm (in l).

4.9 Results for the finite case

The following table is based on a computational verification of hypothesis $\mathcal{H}(k, s)$. The rows in the table correspond to values of k. The columns correspond to various vales of s. The (k, s)-th entry of the table is the number of symmetric boolean functions f such that $A_{k,s}\nu(f)$ is a constant vector. Hence, whenever this entry is 4, $\mathcal{H}(k, s)$ is true. The least value of $\frac{s+1}{l+s+1}$ for which $\mathcal{H}(l + s, s)$ is true in this table is for $l = 18, s = 2$, giving the ratio 3/19. This table is also our reason to conjecture that for all k, $\mathcal{H}(k, 4)$ is true!

k	s=1	s=2	s=3	s=4
2	4	-	-	-
3	6	4	-	-
4	8	4	4	-
5	8	4	4	-
6	20	4	4	4
7	26	8	4	4
8	48	10	6	4
9	42	10	6	4
10	64	6	6	4
11	66	4	4	4
12	144	4	4	4
13	178	8	4	4
14	452	14	6	4
15	428	26	8	4
16	576	12	12	4
17	514	4	4	4
18	1072	4	4	4
19	1442	12	4	4
20	2864	16	8	4
21	2534	16	8	4
22	4608	8	8	4
23	6402	8	4	4
24	12448	10	6	4
25	9350	22	6	4
26	-	-	-	4
27	-	-	-	4
28	-	-	-	4
29	-	-	-	4

CHAPTER V

HARDNESS OF LATTICE PROBLEMS

5.1 Introduction

An n-dimensional lattice $\mathcal{L}$ is a set of vectors $\{\sum_{i=1}^{n} a_i \mathbf{b}_i \mid a_i \in \mathbb{Z}\}$, where $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n \in \mathbb{R}^n$ is a set of independent vectors, called the *basis* for the lattice. The same lattice could have many bases. Given a basis for an n-dimensional lattice, the Shortest Vector Problem asks for the shortest non-zero vector in the lattice. The length of the vectors can be measured in any ℓ_p norm ($p \geq 1$), and the corresponding optimization problem is denoted by SVP_p.

For a thorough treatment of the algorithmic theory of lattices, we refer to Micciancio and Goldwasser's book [86]. The Shortest Vector Problem has been studied since the time of Gauss ([43], 1801), who gave an algorithm for SVP_2 in 2-dimensions. The general problem for arbitrary dimensions was formulated by Dirichlet in 1842. The theory of Geometry of Numbers by Minkowski [87] deals with the existence of shortest non-zero vectors in lattices. In a celebrated result, Lenstra, Lenstra and Lovász [72] gave a polynomial time algorithm for approximating SVP_2 within factor $2^{n/2}$. This algorithm has numerous applications, e.g. factoring rational polynomials [72], breaking knapsack-based cryptosystems [69], checking the solvability by radicals [70] and integer programming in a fixed number of variables ([72], [73], [56]). Further, Schnorr [108] improved the approximation factor to a sub-exponential: $2^{\frac{n(\log\log n)^2}{\log n}}$. Since all ℓ_p norms are within factor $\sqrt{n}$ from the ℓ_2 norm, these algorithms give similar approximations for SVP_p for any p. It is a major open problem whether SVP_2 has polynomial factor approximations that run in polynomial time. Exact computation of SVP_2 in exponential time is also investigated, for instance in [57], [7].

In 1981, van Emde Boas [118] proved that SVP_∞ is NP-hard and conjectured that the same is true for any ℓ_p norm. However proving NP-hardness for any finite p (in particular $p = 2$) was an embarrassing open problem for long time. A breakthrough result by Ajtai [5] in 1998 finally showed that SVP_2 is NP-hard. This result was strengthened by Micciancio

[84], who showed that SVP_p is hard to approximate within some constant factor, specifically, $2^{1/p} - \delta$ for every $\delta > 0$. Recently, Khot [62] showed that for every $\delta > 0$, there is a constant $p(\delta)$ such that for $p \geq p(\delta)$, SVP_p is hard to approximate within factor $p^{1-\delta}$. All these reductions are randomized reductions.

Showing hardness of approximation results for SVP_2 was greatly motivated by Ajtai's discovery [4] of worst-case to average-case hardness, and subsequent construction of lattice-based public key cryptosystem by Ajtai and Dwork [6]. Ajtai showed that if there is a randomized polynomial time algorithm for solving SVP_2 on a non-negligible fraction of lattices from a certain natural class of lattices, then there is a randomized polynomial time algorithm for approximating SVP_2 on *every* instance within some polynomial factor n^c. Ajtai-Dwork's work gave hope, for the first time, that cryptography could be based on the (conjectured) worst-case hardness of a problem. Their work implies that if n^c-approximation to SVP_2 is hard, then one can construct a secure cryptosystem. The constant c was noted to be 19 in [26]. Related results and improvements were obtained in [27, 85].

Unfortunately, there are barriers to showing strong hardness results. In fact, showing factor n NP-hardness for SVP_2 would imply that $\mathsf{NP} = \mathsf{coNP}$ [68] and showing factor $\sqrt{n/O(\log n)}$ NP-hardness would imply that $\mathsf{coNP} \subseteq \mathsf{AM}$ [44], which is not true, unless the polynomial hierarchy collapses to Σ_2 [23]. (For general ℓ_p norms we know this only for $O(n)$ [44].)

5.1.1 Our results

We obtain new hardness results for hardness of approximation of SVP_p. This is an improvement over previous results for certain range of values of p. We show that:

Theorem 5.1.1. *For $p \geq 46$, it is hard to approximate SVP_p (in polynomial time) within a factor of $(4/3)^{1-45.7/p}$, unless $\mathsf{NP} \subseteq \mathsf{ZPP}$.*

Theorem 5.1.2. *For $p \geq 112$, it is hard to approximate SVP_p (in polynomial time) within a factor of $(3/2)^{1-111.7/p}$, unless $\mathsf{NP} \subseteq \mathsf{ZPP}$.*

The cryptosystems of Ajtai-Dwork, which was further improved by Regev in [100], are

actually based on the hardness of approximating the Unique Shortest Vector Problem (denoted USVP). This is a variant of SVP, where in the exact version, we are guaranteed to have (upto sign) only one non zero shortest vector in the lattice. In the approximate version, the only lattice vectors of length at most n^c times the shortest vector are the ones parallel to it. The exact version of USVP_2 was proved to be NP-Hard (under randomized reductions) by Kumar and Sivakumar[66], *a la* Valiant-Vazirani [117]. We extend their result for every ℓ_p norm.

Theorem 5.1.3. *For all $p \geq 1$, USVP_p is hard to compute exactly in polynomial time, unless* $\mathsf{NP} \subseteq \mathsf{BPP}$.

Significance of the hardness of approximation results

We give a brief comparison between the results in this thesis and previous results of Ajtai [5], Micciancio [84] and Khot [62].

- Theorems 5.1.1 and 5.1.2 hold for reasonably small and (more importantly) *explicit* values of p. This is an advantage over Khot's result that holds only for huge values of p, which depend on non-explicit constants in Raz's Parallel Repetition Theorem [99].

- Our proofs are very simple (as opposed to Ajtai and Micciancio's proofs). Theorem 5.1.1 is a direct reduction from a result of Holmerin [50] about Independent Sets in 4-uniform hypergraphs [1], where as Theorem 5.1.2 is a reduction from a similar result of Khot [61] about Independent Sets in 3-uniform hypergraphs.

- Like Khot's result, our result works under the assumption $\mathsf{NP} \not\subseteq \mathsf{ZPP}$. Ajtai and Micciancio's results require the assumption $\mathsf{NP} \not\subseteq \mathsf{RP}$.

Techniques and Overview

We give a straightforward reduction from Holmerin's result that can be stated as follows. We are given a 4-uniform hypergraph (V, E), with vertex set V, $|V| = n$ and a set of edges

[1] A hypergraph $\mathcal{H} = (V, E)$ is k-uniform if all its edges contain exactly k vertices. A set $I \subseteq V$ is called *independent* if for all $e \in E$, $|e \cap I| \leq k - 1$, i.e. I does not contain any edge.

E. Then it is hard to distinguish between the following cases : **(Good case)** There is a set $I \subseteq V$ with $|I| = n/2$, such that for all $e \in E$, $|e \cap I| \leq 3$. In other words, there is an independent set of size $n/2$. **(Bad case)** : For any set $I \subseteq V$ of size δn, for at least $\delta^4 - o(1)$ fraction of edges e, we have $|e \cap I| = 4$. In particular, there is no independent set of size about δn.

As described in Section 5.3.1, it is easy to build an SVP_p instance from a hypergraph. The rows of the vertex-edge adjacency matrix of the hypergraph are essentially the basis vectors for the SVP_p instance. In the good case, there is a (short) lattice vector that has all co-ordinates bounded by 3 in magnitude, and in the bad case, there are about δ^4 fraction of co-ordinates, which are at least 4 in magnitude. Thus, we get a non-trivial hardness result, provided $(4/3)^p > 1/\delta^4$. There are a few technicalities that can be handled using ideas from [62]. Theorem 5.1.2 follows from Khot's results on 3-uniform hypergraphs in a similar manner [61].

The proof of the hardness of USVP_p is presented in Section 5.4. We generalize the reduction of [66] so that it does not use any specific property of the ℓ_2 norm. This involves proving a simple but important geometric fact about the number of lattice points in ℓ_p norm at a distance close to the length of the shortest vector.

Note:

Following this work, Khot [63], very recently, showed that for $1 < p < \infty$, SVP is hard to approximate within any constant factor, assuming $\mathsf{NP} \not\subseteq \mathsf{BPP}$, and, hard to approximate within a factor of $2^{\log n^{1/2-\epsilon}}$, assuming $\mathsf{NP} \not\subseteq \mathsf{BPTIME}(2^{\mathrm{polylog} n})$. He uses a very different reduction than all known ones, and boosts the hardness factor via a variant of the tensor product construction for lattices. Our results still have the advantage of being simple and holding under the weaker assumption $\mathsf{NP} \not\subseteq \mathsf{ZPP}$.

5.2 Preliminaries

For any $p \geq 1$, define the ℓ_p norm of a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ to be $\|\mathbf{x}\|_p := (\sum_{i=1}^n |x_i|^p)^{\frac{1}{p}}$. As in [62], we first give an equivalent formulation of the Shortest Vector Problem in ℓ_p norm.

SVP_p

For $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and a set of linear forms $\mathbf{\Phi} := \{\phi_1, \phi_2, \ldots, \phi_t\}$, where

$$\phi_i(\mathbf{x}) := \sum_{j=1}^{n} b_{ij} x_j \qquad b_{ij} \in \mathbb{Z},$$

find a vector $\mathbf{x} \in \mathbb{Z}^n \backslash \{\mathbf{0}\}$ which minimizes $(\sum_{i=1}^{t} |\phi_i(\mathbf{x})|^p)^{1/p}$. [2] Denote this quantity by $\lambda_{1,p}(\mathbf{\Phi})$. In order to show a factor γ-hardness for SVP_p, it suffices to show that it is hard to produce a vector $\mathbf{y} \in \mathbb{Z}^n \backslash \{\mathbf{0}\}$ such that $\|\mathbf{y}\|_p^p \leq \gamma^p \lambda_{1,p}^p(\mathbf{\Phi})$.

5.2.1 Hypergraph independent set problem

A hypergraph $\mathcal{H} = (V, E)$ is k-uniform if all its edges contain exactly k vertices. A set $I \subseteq V$ is called *independent* if for all $e \in E$, $|e \cap I| \leq k - 1$, i.e. I does not contain any edge. For an integer $k > 2$ and $\delta_1, \delta_2, \mu > 0$, consider the promise problem $\mathbf{H}(k, \delta_1, \delta_2, \mu) := \bigcup_{n \geq 0} \mathbf{H}(k, \delta_1, \delta_2, \mu)_n$, where $\mathbf{H}(k, \delta_1, \delta_2, \mu)_n = (\mathrm{YES}(k, \delta_1, \delta_2, \mu)_n, \mathrm{NO}(k, \delta_1, \delta_2, \mu)_n)$, and

- $\mathrm{YES}(k, \delta_1, \delta_2, \mu)_n$ consists of all k-uniform hypergraphs on n vertices that have an independent size of at least $\delta_1 n$.
- $\mathrm{NO}(k, \delta_1, \delta_2, \mu)_n$ consists of all k-uniform hypergraphs on n vertices such that every vertex set of size at least $\delta_2 n$ contains at least μ fraction of its edges.

Holmerin proved the following result in [50].

Theorem 5.2.1. *For every $\epsilon > 0$, for all $\delta > 0$, the promise problem $\mathbf{H}(4, 1/2, \delta, \delta^4 - \epsilon)$ is NP-hard.*

Since the above result is true for all $\epsilon > 0$, for our reduction, we will assume that $\mathbf{H}(4, 1/2, \delta, \delta^4)$ is NP-Hard. Khot [61] proved a similar result about 3-uniform hypergraphs.

Theorem 5.2.2. *For all $\delta > 0$, the promise problem $\mathbf{H}(3, 1/3, \delta, \delta^9/2^{19})$ is NP-hard.*

[2] We must remark here that if b_{ij}-s are not restricted to be in $\mathbb{Q}$, then the set $\{(\phi_1(\mathbf{x}), \ldots, \phi_t(\mathbf{x})) \,|\mathbf{x} \in \mathbb{Z}^n\}$ need not be a lattice. Indeed, it is known that if b_{ij}-s are rational, then this set is a lattice and a basis, called the Hermite Normal Form, can be found in polynomial time. See [86] for details. Hence, in this setting, the above formulation is equivalent to the standard formulation for SVP, where a basis for the lattice is given.

5.3 Reduction

In this section we reduce the promise problem $\mathbf{H}(4, 1/2, \delta, \delta^4)$ to SVP_p. The same reduction (with suitable modifications) applies to $\mathbf{H}(3, 1/3, \delta, \delta^9/2^{19})$. We give the proof for the 4-uniform case. We give hardness result arising from 3-uniform case in Section 5.3.6.

5.3.1 The lattice from the hypergraph

Let $\mathcal{H} = (V, E)$ be a 4-uniform hypergraph with $n := |V|$ and $m := |E|$. We may assume that the vertex set is $\{1, \ldots, n\}$. In ZPP we construct the following instance of SVP_p from $\mathcal{H}$. Pick $a_1, \ldots, a_n$ randomly and independently from the set $\{1, 2, \ldots, \frac{2^{n/2}}{n} - 1\}$. Further let f_I, f_R, f_E be fixed positive functions of n, m and p to be decided later. Think of them as weights to the corresponding linear forms. There are three type of linear forms. All of them are functions from $\mathbb{Z}^n \to \mathbb{R}$.

- **Identity:** For $1 \leq i \leq n$, define ϕ_i as

$$\phi_i(\mathbf{x}) = x_i.$$

- **Random:** $\phi_R(\mathbf{x}) := \sum_{i=1}^n a_i x_i$.

- **Edge:** For each edge $e = \{s, t, u, v\} \in E$ and each of the sixteen vectors $\epsilon = (\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4) \in \{-1, 1\}^4$,

$$\phi_{e,\epsilon}(\mathbf{x}) := \epsilon_1 x_s + \epsilon_2 x_t + \epsilon_3 x_u + \epsilon_4 x_v.$$

Define $\mathbf{\Phi}(\mathcal{H}) := \{\cup_{i=1}^n f_I^{1/p} \phi_i\} \cup \{f_R^{1/p} \phi_R\} \cup \{\cup_{e \in E, \epsilon \in \{-1,1\}^4} f_E^{1/p} \phi_{e,\epsilon}\}$. There are $n + 1 + 16m$ linear forms. These are obtained from the linear forms $\phi_i, \phi_R, \phi_{e,\epsilon}$ by weighting them with $f_I^{1/p}, f_R^{1/p}, f_E^{1/p}$ respectively. If $\lambda_{1,p}(\mathbf{\Phi}(\mathcal{H}))$ is the shortest vector in the corresponding lattice, then

$$\lambda_{1,p}^p(\mathbf{\Phi}(\mathcal{H})) = \min_{\mathbf{x} \in \mathbb{Z}^n \setminus \{\mathbf{0}\}} \left(f_I \sum_{i=1}^n |\phi_i(\mathbf{x})|^p + f_R |\phi_R(\mathbf{x})|^p + f_E \sum_{e,\epsilon} |\phi_{e,\epsilon}(\mathbf{x})|^p \right)$$

Notice that the only part that the reduction uses randomness is in picking a_i-s.

5.3.2 Technical lemmata

For a vector $\mathbf{x}$, define $\mathrm{supp}(\mathbf{x})$ to be the number of non zero entries in $\mathbf{x}$. For $\alpha > 0$, denote by $H(\alpha) := \alpha \log_2(1/\alpha) + (1-\alpha)\log_2(1/(1-\alpha))$. This is the usual binary entropy function. Let $[n] := \{1, \ldots, n\}$.

Lemma 5.3.1. *For all n, $k \geq 2$, for any set of integers $\{a_1, \ldots, a_n\}$ with each $a_i \in \left[\frac{2^{n/k}}{n} - 1\right]$, and for every set $S \subseteq [n]$ with $|S| \geq n/2$, there is a non zero vector $\mathbf{y} \in \{0, 1, -1\}^n$, such that $\sum_{i \in S} a_i y_i = 0$.*

Proof. For $\mathbf{z} = (z_1, \ldots, z_n) \in \{0,1\}^n$, $\left|\sum_{i \in S} a_i z_i\right| \leq \left(\frac{2^{n/k}}{n} - 1\right) n < 2^{n/k}$. The number of such sums is, however, $2^{|S|} \geq 2^{n/2}$. Hence, there are two vector $\mathbf{z}, \mathbf{z}' \in \{0,1\}^n$, such that $\sum_{i \in S} a_i z_i = \sum_{i \in S} a_i z'_i$. Hence, $\sum_{i \in S} a_i (z_i - z'_i) = 0$. Moreover, for all $1 \leq i \leq n$, $z_i - z'_i \in \{-1, 0, 1\}$. So, let $\mathbf{y} := \mathbf{z} - \mathbf{z}'$ to complete the proof. □

Lemma 5.3.2. *For $\delta, \lambda > 0$, $k \geq 2$, if $H(\delta) + \delta + (\lambda + \delta) H\left(\frac{\delta}{\lambda+\delta}\right) < 1/k$, then for large enough n, there is a set $\{a_1, \ldots, a_n\}$ with each $a_i \in \left[\frac{2^{n/k}}{n} - 1\right]$, such that for all vectors $\mathbf{y} \in \mathbb{Z}^n \backslash \{\mathbf{0}\}$, with $\mathrm{supp}(\mathbf{y}) \leq \delta n$ and $\|\mathbf{y}\|_1 \leq \lambda n$, $\sum_{i=1}^n a_i y_i \neq 0$. Moreover, if we pick a_i randomly and independently from the set $\left[\frac{2^{n/k}}{n} - 1\right]$, then, with high probability, the set $\{a_1, \ldots, a_n\}$ satisfies the above property.*

Proof. Pick $a_1, \ldots, a_n$ uniformly and independently from the set $\left[\frac{2^{n/k}}{n} - 1\right]$. Let $\mathbf{y} \in \mathbb{Z}^n \backslash \{\mathbf{0}\}$ with $\mathrm{supp}(\mathbf{y}) \leq \delta n$ and $\|\mathbf{y}\|_1 \leq \lambda n$,

$$\Pr\left[\sum_{i=1}^n a_i y_i = 0\right] \leq \frac{1}{\frac{2^{n/k}}{n} - 1}.$$

But, the number of vectors $\mathbf{y} \in \mathbb{Z}^n$, such that $\mathrm{supp}(\mathbf{y}) \leq \delta n$, and $\|\mathbf{y}\|_1 \leq \lambda n$, is at most

$$2^{\left[H(\delta) + \delta + (\lambda+\delta) H\left(\frac{\delta}{\lambda+\delta}\right)\right] \cdot n}.$$

Hence, by a union bound, we get that the probability that $\sum_{i=1}^n a_i y_i = 0$ for some such $\mathbf{y}$ is at most $\frac{2^{\left[H(\delta) + \delta + (\lambda+\delta) H\left(\frac{\delta}{\lambda+\delta}\right)\right] \cdot n}}{\frac{2^{n/k}}{n} - 1}$. Since $H(\delta) + \delta + (\lambda + \delta) H\left(\frac{\delta}{\lambda+\delta}\right) < 1/k$, for large enough n, we get the desired result. □

Lemma 5.3.3. *Let $x_1, \ldots, x_k$ be k non-zero integers. Let $a_i \in \{1, -1\}$ be chosen randomly and independently. Then, for all p,*

$$E_{a_1,\ldots,a_k}\left[\left|\sum_{i=1}^{k} a_i x_i\right|^p\right] \geq k^p/2^{k-1}.$$

Proof. There is a vector $(b_1, \ldots, b_k) \in \{-1, 1\}^k$ such that, for all $1 \leq i \leq k$, $b_i x_i$ are all positive. Hence, $\left|\sum_{i=1}^{k} b_i x_i\right| = \sum_{i=1}^{k} b_i x_i \geq k$, as x_i-s are non-zero integers. Also, for the vector $(-b_1, \ldots, -b_k)$, $\left|\sum_{i=1}^{k} -b_i x_i\right| = \sum_{i=1}^{k} b_i x_i \geq k$. Thus,

$$E_{a_1,\ldots,a_k}\left[\left|\sum_{i=1}^{k} a_i x_i\right|^p\right] = \frac{1}{2^k} \sum_{(a_1\ldots,a_k)\in\{-1,1\}^k} \left|\sum_{i=1}^{k} a_i x_i\right|^p \geq \frac{1}{2^k}\left(\left|\sum_{i=1}^{k} b_i x_i\right|^p + \left|\sum_{i=1}^{k} -b_i x_i\right|^p\right) \geq \frac{k^p}{2^{k-1}}.$$

This completes the proof of the lemma.

□

5.3.3 YES instances

Let $\mathcal{H} = (V, E)$ be a hypergraph in YES$(4, 1/2, \delta, \delta^4)_n$, i.e. $\mathcal{H}$ is a 4-uniform hypergraph on n vertices with m edges and has an independent size of at least $n/2$. Let $\mathbf{\Phi}(\mathcal{H})$ be the corresponding lattice. The following lemma establishes that in this case the shortest vector cannot be too big.

Lemma 5.3.4. *In the above setting, $\lambda_{1,p}^p(\mathbf{\Phi}(\mathcal{H})) \leq nf_I + f_E\left[4 \cdot 3^p + 12\right] m$.*

Proof. We show this by producing a non zero vector $\mathbf{x} \in \mathbb{Z}$ such that

$$f_I \sum_{i=1}^{n} |\phi_i(\mathbf{x})|^p + f_R|\phi_R(\mathbf{x})|^p + f_E \sum_{e\in E, \epsilon\in\{-1,1\}^4} |\phi_{e,\epsilon}(\mathbf{x})|^p \leq nf_I + f_E\left[4 \cdot 3^p + 12\right] m.$$

By hypothesis, there is an independent set S, with $|S| \geq n/2$. Let $\mathbf{y}$ be a (non zero) vector promised by Lemma 5.3.1 for the set S. Define the vector $\mathbf{x}$ to be $x_j = 0$ for $j \notin S$ and $x_j = y_j$ for $j \in S$. Then it is clear that for all $1 \leq i \leq n$, $|\phi_i(\mathbf{x})| \leq 1$, $\phi_R(\mathbf{x}) = 0$. Now we use the fact that for all edges $e \in E$, $|e \cap S| \leq 3$. Let $e = \{s, t, u, v\}$ be an edge. If $|e \cap S| \leq 2$, then for any $\epsilon \in \{-1, 1\}^4$, $|\phi_{e,\epsilon}(\mathbf{x})| \leq 2$. Hence $\sum_\epsilon |\phi_{e,\epsilon}(\mathbf{x})|^p \leq 16 \cdot 2^p$ ($\leq 4 \cdot 3^p + 12$, for $p \geq 3$). Thus we may assume that $|e \cap S| = 3$, say $s, t, u \in S$. Further we may assume that $x_s = x_t = x_u = 1$. But now there are exactly 4 linear forms which contribute 3^p, while all the others contribute 1. This proves the lemma. □

5.3.4 NO instances

Let $\mathcal{H} = (V, E)$ be a hypergraph in $\mathrm{NO}(4, 1/2, \delta, \delta^4)_n$, i.e. $\mathcal{H}$ is a 4-uniform hypergraph on n vertices with m edges, such that every vertex set of size at least δn contains at least δ^4 fraction of its edges. Again let $\mathbf{\Phi}(\mathcal{H})$ be the corresponding reduction. The following lemma establishes a lower bound on $\lambda_{1,p}(\mathbf{\Phi}(\mathcal{H}))$.

Lemma 5.3.5. *For $p \geq 3$, and for any choice of $\delta, \lambda > 0, f_I, f_E$ which satisfy the following conditions:*

1. $\delta n \, (\lambda/\delta)^p \, f_I \geq \delta n f_I + 2\delta^4 4^p m f_E$
2. $H(\delta) + \delta + (\lambda + \delta) H\left(\frac{\delta}{\lambda+\delta}\right) < 1/2,$

Then, in the setting above, $\lambda_{1,p}^p(\mathbf{\Phi}(\mathcal{H})) \geq \delta n f_I + 2\delta^4 4^p m f_E$.

Proof. We show that for every non zero vector $\mathbf{x} \in \mathbb{Z}^n$,

$$f_I \sum_{i=1}^{n} |\phi_i(\mathbf{x})|^p + f_R |\phi_R(\mathbf{x})|^p + f_E \sum_{e \in E, \epsilon \in \{-1,1\}^4} |\phi_{e,\epsilon}(\mathbf{x})|^p \geq \delta n f_I + 2\delta^4 4^p m f_E. \tag{9}$$

We do this case by case as in [62], but in a simpler way.

Small support and large ℓ_1 norm

Consider any non-zero $\mathbf{x} \in \mathbb{Z}^n$ with $\mathrm{supp}(\mathbf{x}) \leq \delta n$ and $\|\mathbf{x}\|_1 \geq \lambda n$. For any such vector $\mathbf{x}$, $\|\mathbf{x}\|_p$ is minimized when all its non zero coordinates are of the same magnitude. Thus the minimum is attained when each of the δn non-zero coordinates equals λ/δ. It follows from Condition (1) in the lemma that

$$f_I \sum_{i=1}^{n} |\phi_i(\mathbf{x})|^p \geq f_I \delta n \, (\lambda/\delta)^p \geq \delta n f_I + 2\delta^4 4^p m f_E$$

Small support and small ℓ_1 norm

Consider all $\mathbf{x}$ with $\mathrm{supp}(\mathbf{x}) \leq \delta n$ and $\|\mathbf{x}\|_1 \leq \lambda n$. Choose $f_R := \delta n f_I + 2\delta^4 4^p m f_E$. Then by Condition (2) in the lemma and Lemma 5.3.2, with high probability, $\phi_R(\mathbf{x}) \neq 0$ for all such $\mathbf{x}$. Since $\phi_R(\mathbf{x})$ is an integer, $\phi_R(\mathbf{x}) \geq 1$. Hence $f_R |\phi_R(\mathbf{x})|^p \geq \delta n f_I + 2\delta^4 4^p m f_E$, as desired.

Large support

Now we are left with vectors with support at least δn. Let $\mathbf{x}$ be such a vector and $S \subset [n]$ be the indices corresponding to its support. Since $|S| \geq \delta n$, $\sum_{i=1}^{n} |\phi_i(\mathbf{x})|^p \geq \delta n$. Moreover by the property of $\mathcal{H}$, at least $\delta^4 m$ edges of $\mathcal{H}$ lie entirely in S. This fact combined with Lemma 5.3.3 yields that $\sum_{e \in E, \epsilon \in \{-1,1\}^4} |\phi_{e,\epsilon}(\mathbf{x})|^p \geq \delta^4 m \cdot 2 \cdot 4^p$

This completes the proof of the Lemma. □

5.3.5 Hardness factor and choosing the parameters

For $p \geq 3$, for any choice of $\delta, \lambda > 0, f_I, f_E$ which satisfy the following:

1. $\delta n \, (\lambda/\delta)^p \, f_I \geq \delta n f_I + 2\delta^4 4^p m f_E$
2. $H(\delta) + \delta + (\lambda + \delta) H\left(\frac{\delta}{\lambda+\delta}\right) < 1/2,$

Lemmata 5.3.4 and 5.3.5 imply that SVP_p is hard to approximate within a factor of

$$\left(\frac{\delta n f_I + 2\delta^4 4^p m f_E}{n f_I + f_E \left[4 \cdot 3^p + 12\right] m} \right)^{1/p}.$$

Choose f_I, f_E such that $n f_I = f_E[4 \cdot 3^p + 12]m$. Choose $\lambda = 4\delta/3$ so that Condition (1) is satisfied. With this choice of λ, it can be verified that Condition (2) is satisfied with $\delta = 0.056$. The hardness factor is at least

$$\left(\frac{2\delta^4 4^p m f_E}{2 f_E [4 \cdot 3^p + 12] m} \right)^{1/p} \geq \left(\frac{\delta^4}{5} (4/3)^p \right)^{1/p} \geq (4/3)^{1 - 45.7/p}$$

Thus we get non-trivial hardness result for $p \geq 46$ and the hardness factor approaches $4/3$ as p grows. This proves Theorem 5.1.1.

5.3.6 Reduction from 3-uniform hypergraphs

We do a similar reduction. The integers a_i for the random linear form are chosen from range $[\frac{2^{n/3}}{n} - 1]$. We skip the details and just give the calculations.

Suppose the following conditions are satisfied,

1. $\delta n (\lambda/\delta)^p f_I \geq \delta n f_I + 3^p \delta^9 / 2^{18} m f_E$
2. $H(\delta) + \delta + (\lambda + \delta) H(\frac{\delta}{\lambda+\delta}) < 1/3$

Then SVP_p is hard to approximate within a factor of

$$\left(\frac{\delta n f_I + 3^p \delta^9/2^{18} m f_E}{n f_I + f_E[4 \cdot 2^p] m}\right)^{1/p}$$

Choose f_I, f_E such that $nf_I = f_E[4 \cdot 2^p]m$. Choose $\lambda = 3\delta/2$ so that Condition (1) is satisfied. Condition (2) is satisfied with $\delta = 0.033$. The hardness factor is at least

$$\left(\frac{3^p \delta^9/2^{18} m f_E}{2 f_E[4 \cdot 2^p] m}\right)^{1/p} = \left(\frac{\delta^9}{2^{21}}(3/2)^p\right)^{1/p} \geq (3/2)^{1-111.7/p}$$

Thus we get non-trivial hardness for $p \geq 112$ and the hardness factor approaches $3/2$ as p grows. This proves Theorem 5.1.2.

5.4 Hardness of Unique Shortest Vector Problem in ℓ_p norm

For $p \geq 1$, the Unique Shortest Vector Problem (USVP) is the following promise problem:

USVP_p Given a set of linearly independent vectors $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \in \mathbb{Z}^n$, and a rational number $r > 0$. Let $\mathcal{L}$ be the lattice generated by $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$.

- **YES** Instances

  $(\mathcal{L}, r)$: $\mathcal{L}$ has exactly 2 non-zero vectors: $(\mathbf{v}, -\mathbf{v})$, of ℓ_p norm less than r.

- **NO** Instances

  $(\mathcal{L}, r)$: $\mathcal{L}$ has no non-zero vector of ℓ_p norm less than r.

It was shown by Kumar and Sivakumar [66] that USVP_2 is NP-Hard under randomized reductions. We show that USVP_p is NP-Hard (under randomized reductions) for all $p \geq 1$. The technique we use here is a generalization of the one used by them. As in [66], we reduce the following promise problem to USVP_p, for $p \geq 1$. Let $1 < \zeta < \sqrt{2}$ be fixed.

PSVP_p

Given a set of linearly independent vectors $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \in \mathbb{Z}^n$, let $\mathcal{L}$ be the lattice generated by them. Notice that all non-zero vectors in this lattice will have norm (for any ℓ_p) at least one.

- **YES** Instances

  $\mathcal{L}$: $\mathcal{L}$ has a non-zero vector with length less than ζ.

- **NO** Instances

  $\mathcal{L}$: All non-zero vectors in $\mathcal{L}$ have length at least ζ.

This is known to be NP-Hard under randomized reductions [5, 28, 84] for all $p \geq 1$. Before we proceed to prove Theorem 5.1.3, we need some notation.

Preliminaries

By $\mathbb{Z}_3 = \{0, 1, 2\}$ we mean the usual field modulo 3. For $s \neq 0 \in \mathbb{Z}_3$, let s^{-1} denote its inverse (under multiplication mod 3). Let $\mathcal{L}$ be the lattice generated by a basis $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$. For a vector $\mathbf{v} \in \mathcal{L}$, written as $\sum_j v_j \mathbf{b}_j$, $\chi(\mathbf{v}) := (v_1 \bmod 3, \ldots, v_n \bmod 3)$. We call this the *characteristic* vector of $\mathbf{v}$. This depends on the choice of a basis for $\mathcal{L}$. Although $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ is a special basis given to us as in input to PSVP_p, with abuse of notation, we will use $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ to refer to the basis of the lattice in discussion. This will be when there is no confusion.

First we need to prove a geometrical fact about vectors of short length in a lattice.

Lemma 5.4.1. *Let $\mathcal{L}$ be a lattice in $\mathbf{R}^n$ generated by $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$. Two vectors $\mathbf{u}, \mathbf{v} \in \mathcal{L}$ cannot satisfy all of the following:*

1. $\mathbf{u} \neq \mathbf{v}$.
2. $\chi(\mathbf{u}) = \chi(\mathbf{v})$.
3. $\|\mathbf{u}\|_p, \|\mathbf{v}\|_p < \frac{3}{2}\lambda_{1,p}(\mathcal{L})$.

Proof. Suppose on the contrary that there are such vectors $\mathbf{u}, \mathbf{v} \in \mathcal{L}$. Since $\mathbf{u} \neq \mathbf{v}$, $\mathbf{u} - \mathbf{v} \neq \mathbf{0}$. Also $\chi(\mathbf{u}) = \chi(\mathbf{v})$ implies $\frac{\mathbf{u}-\mathbf{v}}{3} \in \mathcal{L}$. Hence $\left\|\frac{\mathbf{u}-\mathbf{v}}{3}\right\|_p \geq \lambda_{1,p}(\mathcal{L})$. Using triangle inequality we get $\|\mathbf{u}\|_p + \|\mathbf{v}\|_p \geq 3\lambda_{1,p}(\mathcal{L})$. This contradicts the last condition in the hypothesis. Hence the lemma follows. □

We deduce the following corollary which will be used in the reduction. Here $1 < \zeta < \sqrt{2}$ is as in the definition of PSVP_p.

Corollary 5.4.2. *In any lattice $\mathcal{L} \subseteq \mathbf{R}^n$, the number of points in $\mathcal{L}$ of length less than $\zeta \cdot \lambda_{1,p}(\mathcal{L})$ is at most 3^n.*

5.4.1 The reduction

We are ready to give the reduction. Let $\mathcal{L} \subseteq \mathbf{R}^n$ be an instance of PSVP_p. Let t be $2n$.

- Pick uniformly and independently vectors $\mathbf{w}(1), \ldots, \mathbf{w}(t) \in \mathbb{Z}_3^n$.
- Let $\mathcal{L}_0 = \mathcal{L}$.
- For all $0 \leq k < t$, construct a basis of $\mathcal{L}_{k+1}$ from a basis $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ of $\mathcal{L}_k$ and $\mathbf{w}(k+1) = (w_1, \ldots, w_n)$ as follows. (This is not an arbitrary basis of $\mathcal{L}_k$ - but computed recursively from $\mathcal{L}_0$.) Let i be any index such that $w_i \neq 0$. If no such i exists, let $\mathcal{L}_{k+1} = \mathcal{L}_k$. Otherwise let $\mathcal{L}_{k+1}$ be the lattice generated by the basis $\{\mathbf{b}'_1, \ldots, \mathbf{b}'_n\}$ defined as following:

  For $j \neq i$, let $\mathbf{b}'_j := \mathbf{b}_j - w_i^{-1} w_j \mathbf{b}_i$, and let $\mathbf{b}'_i := 3\mathbf{b}_i$.
- Pick a random $j \in \{1, \ldots, t\}$ and output $(\mathcal{L}_j, \zeta)$.

Notice that this reduction is a generalization of [66] and can also be stated for any prime other than 3. Indeed, [66] use the prime 2. For the ℓ_2 norm, Kumar and Sivakumar use the law of parallelograms: for any two vectors $\mathbf{u}$ and $\mathbf{v}$, $2\|\mathbf{u}\|_2^2 + 2\|\mathbf{u}\|_2^2 = \|\mathbf{u} - \mathbf{v}\|_2^2 + \|\mathbf{u} + \mathbf{v}\|_2^2$. Thus, if $\mathbf{u}$ and $\mathbf{v}$ are two distinct vectors with the same characteristic vectors modulo 2, then both $\frac{\mathbf{u}-\mathbf{v}}{2}$ and $\frac{\mathbf{u}+\mathbf{v}}{2}$ are non-zero vectors in the lattice. Hence by the law of parallelograms, one of $\mathbf{u}$ or $\mathbf{v}$ has to be of length at least $\sqrt{2}$ times the length of a shortest vector in the lattice. Thus, all non-zero vectors in a ball of radius less than $\sqrt{2}$ times the length of the shortest vector, must have different characteristic vectors modulo 2. This gives an upper bound (number of distinct characteristic vectors) on the number of non-zero distinct vectors in such a ball around the origin. For $p \neq 2$, the law of parallelograms *does not* hold. Hence, one cannot in general, bound the number of vectors in such a ball around the origin, by just considering the characteristic vector modulo 2. This is the reason we generalize the reduction of [66] to an arbitrary prime. The smallest prime for which this works is 3. The remainder of our proof of hardness of USVP_p follows that of [66] (for $p = 2$) closely. We give the details in the next section.

5.4.2 Proofs

We will prove a series of simple facts about the reduction which will establish Theorem 5.1.3. It is obvious that $\mathcal{L} = \mathcal{L}_0 \supseteq \mathcal{L}_1 \supseteq \cdots \supseteq \mathcal{L}_{2n}$. Hence we get the following fact.

Fact 5.4.3. *If $\mathcal{L}$ is a NO instance of PSVP_p, then $(\mathcal{L}_j, \zeta)$ is a NO instance of USVP_p with probability* 1.

Fact 5.4.4. *Let $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ be a basis for $\mathcal{L}_k$ and $\mathbf{v} = \sum_j v_j \mathbf{b}_j \in \mathcal{L}_k$ with $\chi(\mathbf{v}) \neq \mathbf{0}$. Let $\{\mathbf{b}'_1, \ldots, \mathbf{b}'_n\}$ be the basis for $\mathcal{L}_{k+1}$ generated as in the reduction. Then*

$$\Pr[\mathbf{v} \in \mathcal{L}_{k+1}] = \frac{1}{3}.$$

Proof. Let i be the index chosen in the reduction to select w_i, and hence define $\mathcal{L}_{k+1}$. Write $\mathbf{v}$ in the basis for $\mathcal{L}_{k+1}$ we get $\mathbf{v} = \sum_{j \neq i} v_j \mathbf{b}'_j + \left(\sum_{j \neq i} v_j w_i^{-1} w_j - 2v_i\right) \mathbf{b}_i$. Hence $\mathbf{v} \in \mathcal{L}_{k+1}$ iff $\sum_{j \neq i} v_j w_i^{-1} w_j - 2v_i \equiv 0 \bmod 3$. This is exactly the event that $\langle \chi(\mathbf{v}), \mathbf{w}(k+1) \rangle = 0$. This happens with probability exactly 1/3. (Here we used the fact that $w_i \not\equiv 0 \bmod 3$.) □

Fact 5.4.5. *Let $\mathbf{v}$ be a non-zero vector in $\mathcal{L}$ with basis $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$, such that $\|\mathbf{v}\|_p < \zeta \cdot \lambda_{1,p}(\mathcal{L})$, then $\chi(\mathbf{v}) \neq \mathbf{0}$.*

Proof. Assume the contrary. Since $\chi(\mathbf{v}) = \mathbf{0}$, $\mathbf{u} := \mathbf{v}/3 \in \mathcal{L} \backslash \{\mathbf{0}\}$. But then $\mathbf{u}$ is non-zero vector in $\mathcal{L}$ with length less than $\frac{\zeta}{3} \cdot \lambda_{1,p}(\mathcal{L})$ which is a contradiction as $\zeta < \sqrt{2}$. □

Recall that $\mathcal{L} = \mathcal{L}_0$ is a YES instance of PSVP_p. For $0 \leq k \leq t$, let S_k denote the set of non-zero points in $\mathcal{L}_k$ of length less than $\zeta \cdot \lambda_{1,p}(\mathcal{L})$. By definition, $S_0 \supseteq S_1 \supseteq \cdots \supseteq S_t$. By Corollary 5.4.2, $|S_0| \leq 3^t$. Since $\mathcal{L}$ is a YES instance, $S_0 \neq \Phi$. Facts 5.4.5 and 5.4.4 combined now imply the following:

Fact 5.4.6. *For all $0 \leq k < t$, and all $\mathbf{u} \in S$,*

$$\Pr[\mathbf{u} \in S_{k+1} | \mathbf{u} \in S_k] = \frac{1}{3}.$$

The proof of Fact 5.4.4 implies that for any $\mathbf{u} \in S_k$, its inclusion in S_{k+1} depends just on its characteristic vector. Hence the following fact.

Fact 5.4.7. *For all $0 \leq k < t$, and all $\mathbf{u} \neq \mathbf{v} \in S_k$, the events $\mathbf{u} \in S_{k+1}$ and $\mathbf{v} \in S_{k+1}$ are statistically independent.*

Proof. It follows from Lemma 5.4.1 applied to $\mathcal{L}_k$, that $\chi(\mathbf{u}) \neq \chi(\mathbf{v})$. (Here $\chi(\cdot)$ are with respect to the basis of $\mathcal{L}_k$.) But the inclusion of the vectors $\mathbf{u}, \mathbf{v}$ in S_{k+1} depends only on $\chi(\mathbf{u})$ and $\chi(\mathbf{v})$ respectively. These being different, the events are statistically independent. □

A similar argument proves the following fact.

Fact 5.4.8. *For all $0 \leq k \neq l < t$, and all $\mathbf{u} \in S_0$, the events $(\mathbf{u} \in S_{k+1})|(\mathbf{u} \in S_k)$ and $(\mathbf{u} \in S_{l+1})|(u \in S_l)$ are statistically independent.*

Facts 5.4.6 and 5.4.8 immediately imply the following.

Fact 5.4.9. *$S_t \neq \Phi$ with probability at most $3^n 3^{-t} = 3^{-n}$.*

Now we prove that the reduction maps YES instances of PSVP_p to YES instances of USVP_p with inverse polynomial probability. (This can be amplified to any desired value.)

Fact 5.4.10. *If $\mathcal{L}$ is a YES instance of PSVP_p, then with probability $\Omega(n^{-1})$, over the randomization in the reduction, $(\mathcal{L}_j, \zeta)$ is a YES instance of USVP_p.*

Proof. First we may assume that in the reduction $\mathbf{w}(k) \neq \mathbf{0}$ for all k. This is because this happens with negligible probability. By Fact 5.4.9, S_t is empty with probability at least $1 - 3^{-n}$. Since this probability is negligible, we may assume $S_t = \Phi$. Since $S_0 \neq \Phi$, there is a largest index $0 \leq k < t$ such that S_k has at least 2 elements, $\mathbf{u} \neq \mathbf{v}$. By choice of k, $|S_{k+1}|$ is either 0 or 1. We show that with probability at least $1/2$, $|S_{k+1}| = 1$. By Fact 5.4.7 $\mathbf{u} \in S_{k+1}$ and $\mathbf{v} \in S_{k+1}$ are independent. Hence that exactly one of them belongs to S_{k+1} given that $|S_{k+1}| < 2$, is $\frac{4/9}{8/9} = 1/2$. Moreover our reduction picks $k+1$ with probability $\frac{1}{2n}$. Hence the fact follows. □

The proof of Theorem 5.1.3 now follows from Facts 5.4.10 and 5.4.3.

CHAPTER VI

DETERMINISTIC EXTRACTION

6.1 Deterministic extraction

The problem of extracting randomness from *weak* or *imperfect* random sources has many applications in theoretical computer science [93, 110]. The problem is to find an explicit function which transforms a weak-random source into an almost random one. For many naturally arising weak sources, it is known that deterministic functions are not sufficient to extract even one bit [93]. Hence a random seed is necessary. One important family of this type is the set of sources of *min-entropy* k. This is essentially the family of random variables, each uniformly distributed over some subset of strings of $\{0,1\}^n$ of size 2^k. These are the most important random sources from the point of view of complexity. After a long line of incredible research [110], optimal extractors were constructed for this family recently by Lu, Reingold, Vadhan and Wigderson [81].

The original motivation for constructing extractors was to extract high quality random bits from *slightly defective natural* random sources. Of course without using any auxiliary randomness. Initiated by von Neumann [122], many researchers [42, 19, 107, 33] were successful in this. They got around the problem of auxiliary randomness by requiring more than one independent (or *weakly dependent*) sample. Recently, Trevisan and Vadhan [115] revisited the question of studying weak sources for which one can extract deterministically. They showed that under certain complexity theoretic assumptions, there are efficient deterministic extractors for *efficiently samplable weak sources.* They suggested to identify families of weak random sources for which efficient deterministic extraction is possible.

The family of oblivious bit fixing sources (OBFS) constitutes an important set of weak random sources for which (unconditional) deterministic extraction is possible [34]. [1] An

[1] Since this family is efficiently samplable in the notation of [115], one can apply the results of Trevisan and Vadhan to get efficient extractors from them. But these results are conditional.

(n, k)-OBFS is a random source which has k uniform random bits and the other are fixed independently of the free bits. An efficient deterministic extractor for $(n, n/2)$-OBFS was given in [34]. Recently, Kamp and Zuckerman [55] gave a deterministic algorithm to extract almost random bits from a $(n, n^{1/2+\gamma})$-OBFS.

Here we show how to extract almost random bits when the number of free bits can be arbitrarily small [2]. As in [55], the extractor we construct has applications in cryptography. We discuss this in the next section.

Incidentally, our extractor construction also leads to a construction of almost independent set of random variables (in the spirit of Naor and Naor [92]) deriving their randomness from a OBFS. This seems to be of independent interest. This raises the question of whether simple (possibly deterministic) construction of almost independent family of random variables is possible from more general weak sources. This would be of practical interest to the cryptographic community [67].

6.2 Cryptographic motivation

Exposure Resilient Cryptography

Most of cryptography is based on the assumption that the secret key is hidden from the adversary. In the modern context, where the keys are stored on computers, which are in turn connected to huge networks, it becomes necessary to re-evaluate this assumption. Classic solutions for this problem were based on secret sharing or using specialized hardware. But these are costly solutions. A recent branch of cryptography known as *Exposure Resilient Cryptography*, aims to understand and provide more efficient solutions to problems arising when the adversary has access to *most* of the secret key. This was initiated by Rivest [104], who introduced a class of randomized mappings called *All-Or-Nothing Transforms* (AONT). Such a transformation has the property that it is easy to invert given the entire output, but hard to invert even when a small part of the output is not known. This was furthered by Boyko [24], who gave a construction of AONT assuming the random oracle hypothesis.

Canetti *et al.* [29] gave the first construction in the standard computational setting.

[2] Similar results were obtained in [55].

They achieved this by constructing a more powerful primitive: *Exposure Resilient Function* (ERF). This efficiently computable (deterministic) function has the property that the output seems random even if all the bits of the input are revealed. The construction of AONTs was hence reduced to construction of ERFs.

Dodis *et al.* [40] extended this when the adversary can adaptively decide to expose the secret. They gave probabilistic constructions of these adaptively secure ERFs. Kamp and Zuckerman [55] were the first to provide deterministic solution to this problem. Their construction gives a stronger primitive: *Almost Perfect Resilient Function* (APRF). Such a function works even when the adversary can actually fix some of the bits rather than just observing them. Resilient functions were introduced in [34] and almost resilient functions were introduced by [67]. Kamp and Zuckerman constructed APRFs under the assumption that adversary does not have access to at least $n^{1/2+\gamma}$ bits.

The deterministic extractor we construct immediately implies a deterministic APRFs even when the adversary can fix all but n^{ϵ} bits, for any $\epsilon > 0$. This improves on the result of [55].

6.3 Our contribution

Results

We construct a *very simple* and *efficient* extractor for any (n, k)-OBFS. Though our technique works for any k, we state here the most important case:

> For all $0 < \epsilon < 1$, and $m = \Omega(\log n)$, there is an explicit function $\text{Ext} : \{0,1\}^n \to \{0,1\}^m$, such that for any (n, n^{ϵ})-OBFS X, $Ext(X)$ is $1/2^{n^{\Omega(\epsilon)}}$ close to the uniform distribution on $\{0,1\}^m$.

As described above, this immediately implies efficient deterministic construction of k-APRFs for $k = n^{\epsilon}$ for any $\epsilon > 0$.

Another interesting result that we observe is an efficient deterministic construction of a large almost l-wise independent family from any (n, k)-OBFS X. That is, there is a set

of deterministic functions $Y_1, \ldots, Y_t$ on X such that any l of them are almost independent. We can achieve roughly $t = k^{1/2l}/\log^2 k$.

Techniques and overview

Let $(x_1, \ldots, x_n)$ be a sample from an (n, k)-OBFS X. Let M be a positive integer, which is $o(\sqrt{k})$. The extractor is simply $x_1 + \cdots + x_n \bmod M$. To show that this works, we need to prove some results about periodic sums of binomial coefficients. This is done by analyzing certain exponential sums. We present these proofs in Section 6.5.2. We mention that though we proved these results independently, they have been discovered many times, and seem to be more than a century old [45]. We prove the correctness of this extractor in Section 6.6.

The construction of almost independent family of random variables follows from combining this result with the Chinese Remainder Theorem. This is described in detail in Section 6.7.

6.4 Bit Fixing Sources, Extractors, and Exposure Resilient Cryptography

Basic notation

We use the following standard notations. Denote by $\mathcal{U}_n$ the uniformly distributed random variable on the set $\{0,1\}^n$ of n-bit strings. Denote by U_n to be the uniformly distributed random variable on the set $\{0, 1, \ldots, n-1\}$. For two random variables X and Y, we use $\Delta(X, Y)$ to denote the *statistical distance* between X and Y.

Definition 6.4.1. *Let X and Y be two random variables over a discrete set $\mathcal{S}$. The* statistical distance *between X and Y is defined as*

$$\Delta(X,Y) \;:=\; \max_{S \subseteq \mathcal{S}} \left|\Pr[X \in S] - \Pr[Y \in S]\right| \;=\; \frac{1}{2} \cdot \sum_{x \in \mathcal{S}} \left|\Pr[X = x] - \Pr[Y = x]\right|.$$

We say that X and Y are ϵ-close if $\Delta(X, Y) \leq \epsilon$. For an integer n, we use notation $[n]$ to denote the set $\{1, \ldots, n\}$. For a random variable X over $\{0,1\}^n$ and a set $S \subseteq [n]$, let X_S denote the (ordered) projection of X on to the bits of S.

There are many notions of *almost independent* sample spaces [92]. Here we use the following:

Definition 6.4.2. *Let $m_1, \ldots, m_t > 0$ be integers. A set of random variables $\{Y_1, \ldots, Y_t\}$, where $Y_i : \{0,1\}^n \to \{0, 1, \ldots, m_i - 1\}$, is said to be (l, μ)-almost independent, if for any $S \subseteq [t]$, with $|S| = l$, the random vector $(Y_j)_{j \in S}$ is μ-close to $(U_{m_j})_{j \in S}$.*

Bit Fixing Sources and APRFs

Definition 6.4.3. *A source X on n bit strings is said to be an (n, k) oblivious bit fixing source (OBFS), if there is set $I \subseteq [n]$, with $|I| = k$, such that $\Delta(X_I, \mathcal{U}_k) = 0$, and $X_{\bar{I}}$ is a fixed string $\sigma \in \{0,1\}^{n-k}$.*

Definition 6.4.4. *A function $f : \{0,1\}^n \to \{0,1\}^m$ is a $k(n)$ almost perfect resilient function (APRF) if, for any setting of $n - k(n)$ bits of the input string to any fixed values, for all $\sigma \in \{0,1\}^m$, the probability (over the random choices of $k(n)$ bits) that f takes on value σ, denoted by p_σ, satisfies:*

$$\left| p_\sigma - \frac{1}{2^m} \right| \leq \frac{\mu(n)}{2^m},$$

for all negligible functions $\mu(n)$.

Extractors

We will use the following definition of extractors.

Definition 6.4.5. *A function* $\mathrm{Ext} : \{0,1\}^n \to \{0,1\}^m$ *is a* (k, ϵ)-extractor *if for every k-OBFS source X,* $\mathrm{Ext}(X)$ *is ϵ-close to $\mathcal{U}_m$.*

The following lemma implies that to construct k-APRFs, it is sufficient to construct extractors for (n, k)-OBFSs. A proof of this can be found in [55].

Lemma 6.4.6. *Any $(k, 2^{-m}\mu(n))$-extractor $f : \{0,1\}^n \to \{0,1\}^m$, where $\mu(n)$ is negligible, is also a k-APRF.*

6.5 Basic tools

In this section we present basic results which will be used in the constructions.

6.5.1 Random variables

Lemma 6.5.1. *Let X be a random variable on $\{0, 1, \ldots, M-1\}$, with $\Pr[X = i] = p_i$. Suppose for all $i, j \in \{0, 1, \ldots, M-1\}$, $|p_i - p_j| \leq \mu$, then $\Delta(X, U_M) \leq M\mu$.*

Proof. Without loss of generality assume $p_0 \geq p_1 \geq \cdots \geq p_{M-1}$. Then $p_1 - 1/M \leq \mu$. Hence by triangle inequality $|p_i - 1/M| \leq |p_i - p_1| + |p_1 - 1/M| \leq 2\mu$. Hence $\frac{1}{2}\sum_{i=0}^{M-1} |p_i - 1/M| \leq M\mu$. □

6.5.2 Binomial coefficients and exponential sums

For a positive integer M, let $\{0,1\}_M := \exp\left(\frac{2\pi\iota}{M}\right) = \cos\frac{2\pi}{M} + \iota \sin\frac{2\pi}{M}$. Here $\iota = \sqrt{-1}$. For $s \in \{0, 1, \ldots, M-1\}$, define

$$\beta_{k,M}(s) := \sum_{j \equiv s \bmod M} \binom{k}{j}.$$

Lemma 6.5.2. *For integers $k \geq M > s \geq 0$,*

$$\beta_{k,M}(s) = \frac{1}{M} \sum_{j=0}^{M-1} \zeta^{-sj}(1 + \zeta^j)^k.$$

Proof. For $0 \leq j \leq M-1$, it is clear that

$$(1 + \zeta^j) = \sum_{s=0}^{M-1} \beta_{k,M}(s)\zeta^{js}.$$

Let $\mathbf{z} := \left(2^k, \ldots, (1+\zeta^j)^k, \ldots, (1+\zeta^{M-1})^k\right)^T$, and $\mathbf{b} := (\beta_{k,M}(0), \ldots, \beta_{k,M}(j), \ldots, \beta_{k,M}(M-1))^T$. Let V_ζ be the $M \times M$ matrix (with rows and columns indexed by $\{0, 1, \ldots, M-1\}$) with $V_\zeta(i,j) := \zeta^{ij}$. (This is a Vandermonde matrix.) Then the above equations can be written as $V_\zeta \mathbf{b} = \mathbf{z}$. The inverse of V_ζ is easily seen to be the matrix V_ζ^{-1}, with $V_\zeta^{-1}(i,j) = \frac{1}{M}\zeta^{-ij}$. Hence $\mathbf{b} = V_\zeta^{-1}\mathbf{z}$. This proves the lemma. □

The next technical lemma will be crucial to prove that the output of our extractor is almost uniformly distributed.

Lemma 6.5.3. *There is a constant $c > 0$ such that for all $s, t \in \{0, 1, \ldots, M-1\}$, and $k \geq M$,*

$$\frac{1}{2^k} |\beta_{k,M}(s) - \beta_{k,M}(t)| \leq c \exp\left(-\frac{k}{M^2}\right).$$

Proof. From Lemma 6.5.2, $|\beta_{k,M}(s) - \beta_{k,M}(t)|$ is equal to

$$\frac{1}{M}\left|\sum_{j=0}^{M-1}\zeta^{-sj}(1+\zeta^j)^k - \sum_{j=0}^{M-1}\zeta^{-tj}(1+\zeta^j)^k\right| = \frac{1}{M}\left|\sum_{j=1}^{M-1}\zeta^{-sj}(1+\zeta^j)^k - \sum_{j=1}^{M-1}\zeta^{-tj}(1+\zeta^j)^k\right|.$$

Also $|1+\zeta^j| = \left|1+\cos\frac{2\pi j}{M}+\iota\sin\frac{2\pi j}{M}\right| = \sqrt{\left(1+\cos\frac{2\pi j}{M}\right)^2+\sin^2\frac{2\pi j}{M}} = \sqrt{2+2\cos\frac{2\pi j}{M}} = 2\cos\frac{\pi j}{M}$. Moreover for $j \in \{1,\ldots,M-1\}$, $\cos\frac{\pi j}{M}$ is maximized for $j=1$. By triangle inequality

$$|\beta_{k,M}(s) - \beta_{k,M}(t)| \leq \frac{1}{M}\sum_{j=1}^{M-1}\left|1+\zeta^j\right|^k\left|\zeta^{-sj}-\zeta^{-tj}\right| \leq \frac{2(M-1)}{M}\left(2\cos\frac{\pi}{M}\right)^k \leq \left(2\cos\frac{\pi}{M}\right)^k.$$

Now using the power series expansion of $\cos\frac{\pi}{M}$, for large M we can approximate it (upto a universal constant) by $1-\frac{1}{2}\frac{\pi^2}{M^2}$. Using $1+x\leq\exp(x)$, we now get the lemma. □

6.5.3 Primes and their distribution

We will need the following case of the Chinese Remainder Theorem.

Lemma 6.5.4. *Let $q_1,\ldots,q_l$ be distinct primes and $b_i \in \{0,1,\ldots,q_i-1\}$ be arbitrary, then the following system of congruences*

$$\begin{aligned} x &\equiv b_1 \mod q_1 \\ x &\equiv b_2 \mod q_2 \\ &\vdots \\ x &\equiv b_l \mod q_l \end{aligned}$$

has exactly one solution modulo the product $q_1q_2\cdots q_l$.

The following is an easy fact and follows from the Prime Number Theorem.

Lemma 6.5.5. *For large enough n, the number of primes between $n/2$ and n is $\Omega\left(\frac{n}{\log^2 n}\right)$.*

6.6 Extractor construction

Now we are ready to describe the extractor. Let X be a (n,k)-OBFS. Let M be a positive integer which is at most k. Consider the following function:

$$\mathrm{Ext}_M(x_1,x_2,\ldots,x_n) := x_1+x_2+\cdots+x_n \mod M.$$

We think of Ext_M is a function from $\{0,1\}^n$ to $\{0,1,\ldots,M-1\}$. One can also think of it as a function to $\{0,1\}^{\lfloor \log M \rfloor}$. We will use both without any confusion.

The following theorem shows that this extremely simple deterministic function is a good extractor.

Theorem 6.6.1. *For all $0 < \epsilon < 1$, for all (n,k)-OBFS X, and $M = O(k^{\frac{1-\epsilon}{2}})$, $\Delta(\mathrm{Ext}_M(X), U_M) \leq O(2^{-k^{\Omega(\epsilon)}})$.*

Proof. We can without loss of generality assume that the fixed bits of X are set to 0. Let $x = (x_1,\ldots,x_n)$ be a sample chosen from the distribution X. It is clear that for $s \in \{0,1,\ldots,M-1\}$,

$$\Pr[\mathrm{Ext}_M(x_1,\ldots,x_n) \equiv s \mod M] = \frac{\sum_{j \equiv s \mod M} \binom{k}{j}}{2^k} = \frac{\beta_{k,M}(s)}{2^k}.$$

The proof is now immediate from Lemmata 6.5.3 and 6.5.1. □

The following is probably the most important case of extraction and hence we mention it explicitly.

Corollary 6.6.2. *For all $0 < \epsilon < 1$, and $m = \Omega(\log n)$, there is an explicit function $\mathrm{Ext} : \{0,1\}^n \to \{0,1\}^m$, such that for any (n, n^ϵ)-OBFS X, $\Delta(\mathrm{Ext}(X), \mathcal{U}_m) \leq 1/2^{n^{\Omega(\epsilon)}}$.*

As a corollary to this corollary we obtain the following.

Corollary 6.6.3. *For any $0 < \epsilon < 1$, there is an efficient n^ϵ-APRF $f : \{0,1\}^n \to \{0,1\}^{\Omega(\log n)}$.*

6.7 Constructing (l,μ)-almost independent family of random variables

In this section we show how to construct simple (l,μ)- independent family of random variables using results proved so far. Let X be a (n,k)-OBFS. Let $0 < \epsilon < 1$. Let $p_1,\ldots,p_t$ be primes such that for any $S \subseteq [t]$, $|S| = l$, $\prod_{j \in S} p_j \leq ck^{\frac{1-\epsilon}{2}}$, for some fixed constant $c > 0$. Let $Y_i(X) := \mathrm{Ext}_{p_i}(X)$. Here Ext is the extractor defined in the previous section. That is for $x = (x_1,\ldots,x_n)$, $Y_i(x) := x_1 + \cdots + x_n \mod p_i$. Choose p_i-s to be in the range

$[\frac{1}{2}ck^{\frac{1-\epsilon}{2l}}, ck^{\frac{1-\epsilon}{2l}}]$. By Lemma 6.5.5 we can choose $t = \Omega\left(\frac{k^{\frac{1-\epsilon}{2l}}}{\log^2 k}\right)$. We have the following theorem.

Theorem 6.7.1. *For any fixed $l > 1$, any $0 < \epsilon < 1$, any (n, k)-OBFS X, the family $\{Y_i\}_{i=1}^t$ defined above, gives a $(l, 2^{-k^{\Omega(\epsilon)}})$-independent family of random variables. We can choose t to be $\Omega\left(\frac{k^{\frac{1-\epsilon}{2l}}}{\log^2 k}\right)$ and each random variable outputs $\Omega(\log k)$ almost random bits.*

Proof. In light of the discussion before the theorem, it is sufficient to prove that for any $S \subseteq [t]$, $|S| = l$, the random vector $(Y_i)_{i\in S}$ is $2^{-k^{\Omega(\epsilon)}}$-close to $(U_{p_i})_{i\in S}$. We will show this using the Chinese Remainder Theorem.

Without loss of generality assume that $S = \{1, 2, \ldots, l\}$. Let $b_i \in \{0, 1, \ldots, p_i - 1\}$ be arbitrary. Let β be the unique solution $(\bmod\ p_1 \cdots p_l)$ to the congruences $z \equiv b_i \bmod p_i$. Hence

$$\Pr[\wedge_{i=1}^{l}(Y_i = b_i)] = \Pr_{x\in X}[x_1 + x_2 + \cdots + x_n \equiv \beta \mod p_1 \cdots p_l].$$

Now we can use the Theorem 6.6.1 with $\mathrm{Ext}_{p_1\cdots p_l}$, to get the result of this theorem. □

REFERENCES

[1] "Mathematical developments arising from hilbert's problems," in *Proc. Symp. Pure Math.* (Browder, F., ed.), no. 28 in American Mathematical Society, 1976.

[2] Agrawal, M. and Biswas, S., "Primality and identity testing via Chinese Remaindering," *Journal of the ACM*, vol. 50, no. 4, pp. 429–443, 2003.

[3] Agrawal, M., Kayal, N., and Saxena, N., "PRIMES is in P." August 2002.

[4] Ajtai, M., "Generating hard instances of lattice problems," in *Proceedings of the ACM Symposium on the Theory of Computing*, no. 28, pp. 99–108, 1996.

[5] Ajtai, M., "The shortest vector problem in ℓ_2 is NP-hard for randomized reductions," in *Proceedings of the ACM Symposium on the Theory of Computing*, no. 30, pp. 10–19, 1998.

[6] Ajtai, M. and Dwork, C., "A public-key cryptosystem with worst-case/average-case equivalence," in *Proceedings of the ACM Symposium on the Theory of Computing*, no. 29, pp. 284–293, 1997.

[7] Ajtai, M., Kumar, R., and Sivakumar, D., "A sieve algorithm for the shortest lattice vector problem," in *Proceedings of the ACM Symposium on the Theory of Computing*, no. 33, pp. 601–610, 2001.

[8] Arora, S., Lund, C., Motwani, R., Sudan, M., and Szegedy, M., "Probabilistic checking of proofs: a new characterization of NP," *Journal of the ACM*, vol. 45, no. 1, pp. 70–122, 1998.

[9] Artin, E., "Über die zerlegung definiter funktionen in quadrate," *Abh. Math. Sem. Univ. Hamburg*, vol. 5, pp. 100–115, 1927.

[10] Babai, L., Fortnow, L., and Lund, C., "Non-deterministic exponential time has twoprover interactive protocols," *Computational Complexity*, vol. 1, no. 1, pp. 3–40, 1991.

[11] Baker, R. C. and Harman, G., "The Brun-Titchmarsh theorem on average," in *Proceedings of the conference in honor of Heini Halberstam*, vol. 138 of *Progresses in Mathematics*, (Boston, MA), pp. 39–103, Birkhäuser, 1996.

[12] Barak, B., Impagliazzo, R., and Wigderson, A., "Extracting randomness from few independent sources," in *Annual Symposium on Foundations of Computer Science*, no. 45, 2004. To appear.

[13] Bernasconi, A., Codenotti, B., and Simon, J., "Fourier transform of boolean functions." Preliminary version available at `http://www.imc.pi.cnr.it/~codenotti/online.html`.

[14] Blekherman, G., "There are significantly more non-negative polynomials than sums of squares." Manuscript, 2004.

[15] Blum, A., "Relevant examples and relevant features: Thoughts from computational learning," in *AAAI Fall Symposium on* Relevance, 1994.

[16] Blum, A., "Open problems: Learning a function of r relevant variables," in *Proceedings of the Annual Conference on Computational Learning Theory*, no. 16 in Lecture Notes in Computer Science, pp. 731–733, Springer, 2003.

[17] Blum, A., Furst, M., Kearns, M., and Lipton, R. J., "Cryptographic primitives based on hard learning problems," in *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference*, no. 773 in Lecture Notes in Computer Science, pp. 278–291, 1994.

[18] Blum, A. and Langley, P., "Selection of relevant features and examples in machine learning," *Artificial Intelligence*, vol. 97, no. 1–2, pp. 245–271, 1997.

[19] Blum, M., "Independent and unbiased coin flips from a correlated biased source- a finite markov chain," *Combinatorica*, vol. 6, no. 2, pp. 97–108, 1986.

[20] Blum, M. and Kannan, S., "Designing programs that check their work," *Journal of the ACM*, vol. 42, no. 1, pp. 269–291, 1995.

[21] Bochnak, H., Coste, M., and Roy, M. F., *Real algebraic geometry.* Springer-Verlag, 1998.

[22] Boneh, D. and Lipton, R. J., "Algorithms for black-box fields and their application to cryptography," in *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference*, no. 1109 in Lecture Notes in Computer Science, pp. 283–297, Springer, 1996.

[23] Boppana, R., Hastad, J., and Zachos, S., "Does co-NP have short interactive proofs?," *Information Processing Letters*, vol. 25, pp. 127–132, 1987.

[24] Boyko, V., "On the security properties of the OAEP as an all-or-nothing transform," in *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference*, vol. 1666 of *Lecture Notes in Computer Science*, pp. 513–518, Springer, 1999.

[25] Bshouty, N., Jackson, J., and Tamon, C., "More efficient PAC learning of DNF with membership queries under the uniform distribution," in *Proceedings of the Annual Conference on Computational Learning Theory*, no. 12 in Lecture Notes in Computer Science, pp. 286–295, Springer, 1999.

[26] Cai, J. Y., "A new transference theorem in the geometry of numbers and new bounds for Ajtai's connection factor," *Discrete Applied Mathematics*, vol. 126, no. 1, pp. 9–31, 2003.

[27] Cai, J. Y. and Nerurkar, A., "An improved worst-case to average-case connection for lattice problems," in *Proceedings of the Annual IEEE Conference on Computational Complexity*, no. 38, pp. 46–57, 1997.

[28] Cai, J. Y. and Nerurkar, A., "Approximating the SVP to within a factor $(1 + 1/\dim^{\epsilon})$ is NP-hard under randomized reductions," *Journal of Computer and System Sciences*, vol. 59, no. 2, pp. 221–239, 1999.

[29] Canett, R., Dodis, Y., Halevi, S., Kushilevitz, E., and Sahai, A., "Exposure-resilient functions and all-or-nothing transforms," in *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques*, vol. 1807 of *Lecture Notes in Computer Science*, pp. 453–469, Springer, 2000.

[30] Chari, S., Rohatgi, P., and Srinivasan, A., "Randomness-optimal unique element isolation with applications to perfect matching and related problems," *SIAM Journal of Computing*, vol. 24, no. 5, pp. 1036–1050, 1995.

[31] Chen, Z. and Kao, M., "Reducing randomness via irrational numbers," *SIAM Journal of Computing*, vol. 29, no. 4, pp. 1247–1256, 2000.

[32] Choi, M. D., Dai, Z. D., Lam, T. Y., and Reznick, B., "The pythagoras number of some affine algebras and local algebras," *J. Reine Angew. Math.*, vol. 336, pp. 45–82, 1982.

[33] Chor, B. and Goldreich, O., "Unbiased bits from sources of weak randomness and probabilistic communication complexity," *SIAM Journal of Computing*, vol. 17, no. 2, pp. 230–261, 1988.

[34] Chor, B., Goldreich, O., Hastad, J., Friedman, J., Rudich, S., and Smolensky, R., "The bit extraction problem of t-resilient functions," in *Annual Symposium on Foundations of Computer Science*, no. 26, pp. 396–407, 1985.

[35] Cook, S., "The complexity of theorem-proving procedures," in *Proceedings of the ACM Symposium on the Theory of Computing*, no. 3, pp. 151–158, 1971.

[36] DeMillo, R. and Lipton, R. J., "A probabilistic remark on algebraic program testing," *Information Processing Letters*, vol. 7, no. 4, pp. 193–195, 1978.

[37] Devanur, N. R., Lipton, R. J., and Vishnoi, N. K., "Learning symmetric juntas." Unpublished, October 2003.

[38] Devanur, N. R., Lipton, R. J., and Vishnoi, N. K., "On the complexity of Hilbert's 17th problem." Manuscript, March 2003.

[39] Dodis, Y., *Exposure Resilient Cryptography*. PhD thesis, 2000, MIT.

[40] Dodis, Y., Sahai, A., and Smith, A., "On perfect and adaptive security in exposure-resilient cryptography," in *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques*, vol. 2045 of *Lecture Notes in Computer Science*, pp. 301–324, Springer, 2001.

[41] Dubois, D. W., "Note on of Hilbert's 17th problem," *Bull. Amer. Math. Soc.*, vol. 73, pp. 540–541, 1967.

[42] Elias, P., "The efficient construction of an unbiased random sequence," *Annals of Math. Stat.*, vol. 42, no. 3, pp. 865–870, 1972.

[43] Gauss, C., *Disquisitiones arithmeticae*. Yale Univ. Pres, 1966. English translation by A. A. Clarke.

[44] Goldreich, O. and Goldwasser, S., "On the limits of non-approximability of lattice problems," in *Proceedings of the ACM Symposium on the Theory of Computing*, no. 30, pp. 1–9, 1998.

[45] Granville, A., "Arithmetic properties of binomial coefficients." Available at `http://www.dms.umontreal.ca/ andrew/Binomial`.

[46] Hilbert, D., "Über die darstellung definiter formen als summen von formenquadraten," *Math. Ann.*, vol. 32, pp. 342–350, 1888.

[47] Hilbert, D., "Über ternäre definite formen," *Acta Math.*, vol. 17, pp. 169–198, 1893.

[48] Hilbert, D., *Grundlagen der Geometrie*, ch. 7. 1899.

[49] Hilbert, D., "Darstellung definiter formen durch quadrate," *Akad. Wiss. Göttingen*, pp. 284–285, 1900.

[50] Holmerin, J., "Vertex cover on 4-regular hyper-graphs is hard to approximate within $2 - \epsilon$," in *Proceedings of the ACM Symposium on the Theory of Computing*, no. 34, pp. 544–552, 2002.

[51] Ibarra, O. H. and Moran, S., "Probabilistic algorithms for deciding equivalence of straight-line programs," *Journal of the ACM*, vol. 30, no. 1, pp. 217–228, 1983.

[52] Impagliazzo, R. and Wigderson, A., "P=BPP if E requires exponential circuits: Derandomizing the XOR Lemma," in *Proceedings of the ACM Symposium on the Theory of Computing*, no. 29, pp. 220–229, 1997.

[53] Jackson, J., "An efficient membership-query algorithm for learning dnf with respect to the uniform distribution," *Journal of Computer and System Sciences*, vol. 55, pp. 414–440, 1997.

[54] Kabanets, V. and Impagliazzo, R., "Derandomizing polynomial identity tests means proving circuit lower bounds," in *Proceedings of the ACM Symposium on the Theory of Computing*, no. 35, pp. 355–364, 2003.

[55] Kamp, J. and Zuckerman, D., "Deterministic extractors for bit-fixing sources and exposure-resilient cryptography," in *Annual Symposium on Foundations of Computer Science*, no. 44, pp. 91–101, 2003.

[56] Kannan, R., "Improved algorithms for integer programming and related lattice problems," in *Proceedings of the ACM Symposium on the Theory of Computing*, no. 15, pp. 193–206, 1983.

[57] Kannan, R., "Minkowski's convex body theorem and integer programming," *Mathematics of Operations Research*, vol. 12, pp. 415–440, 1987.

[58] Karp, R. M. and Lipton, R. J., "Turing machines that take advice," *Enseign. Math.*, no. 28, pp. 191–201, 1982.

[59] Karp, R., *Reducibility among combinatorial problems*, pp. 85–103. Plenum Press, 1972.

[60] Kearns, M., "Efficient noise-tolerant learning from statistical queries," *Journal of the ACM*, vol. 45, no. 6, pp. 983–1006, 1998.

[61] Khot, S., "Hardness results for coloring 3-colorable 3-uniform hypergraphs," in *Annual Symposium on Foundations of Computer Science*, no. 43, pp. 23–32, 2002.

[62] Khot, S., "Hardness of approximating the shortest vector problem in high ℓ_p norms," in *Annual Symposium on Foundations of Computer Science*, no. 44, pp. 290–297, 2003.

[63] Khot, S., "Hardness of approximating the shortest vector problem in lattices," in *Annual Symposium on Foundations of Computer Science*, no. 45, 2004. To appear.

[64] Khot, S. and Vishnoi, N. K., "Hardness of lattice problems in ℓ_p norm." Manuscript, November 2003.

[65] Klivans, A. and Spielman, D., "Randomness efficient identity testing of multivariate polynomials," in *Proceedings of the ACM Symposium on the Theory of Computing*, no. 33, pp. 216–223, 2001.

[66] Kumar, R. and Sivakumar, D., "On the unique shortest lattice vector problem," *Theoretical Computer Science*, vol. 255, pp. 641–648, 2001.

[67] Kurosawa, K., Johansson, T., and Stinson, D. R., "Almost k-wise independent sample spaces and their cryptographic applications," *Journal of Cryptology*, vol. 14, no. 4, pp. 231–253, 2001.

[68] Lagarias, J., Lenstra, H., and Schnorr, C., "Korkine-Zolotarev bases and successive minima of a lattice and its reciprocal lattice," *Combinatorica*, vol. 10, pp. 333–348, 1990.

[69] Lagarias, J. and Odlyzko, A., "Solving low-density subset sum problems," *Journal of the ACM*, vol. 32, no. 1, pp. 229–246, 1985.

[70] Landau, S. and Miller, G., "Solvability of radicals is in polynomial time," *Journal of Computer and System Sciences*, vol. 30, no. 2, pp. 179–208, 1985.

[71] Lautemann, C., "BPP and the polynomial time hierarchy," *Information Processing Letters*, no. 17, pp. 215–218, 1983.

[72] Lenstra, A., Lenstra, H., and Lovász, L., "Factoring polynomials with rational coefficients," *Math. Ann.*, vol. 261, pp. 513–534, 1982.

[73] Lenstra, H., "Integer programming with a fixed number of variables," Tech. Rep. 81-03, Univ. of Amsterdam, Amsterdam, 1981.

[74] Levin, L. A., "Universal'nye perebornye zadachi (Universal search problems: in Russian," *Problemy Peredachi Informatsii*, vol. 9, no. 3, pp. 265–266, 1973.

[75] Lewin, D. and Vadhan, S., "Checking polynomial identities over any field: Towards a derandomization?," in *Proceedings of the ACM Symposium on the Theory of Computing*, no. 30, pp. 438–447, 1998.

[76] Linnik, Y. V., "On the least prime in an arithmetic progression, I. the basic theorem; II. the Deuring-Heilbronn's phenomenon," *Rec. Math. (Mat. Sbornik)*, vol. 15, no. 57, pp. 139–178 and 347–368, 1944.

[77] Lipton, R. J., *New Directions in Testing*, pp. 191–202. No. 2 in DIMACS Series on Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, 1991.

[78] Lipton, R. J. and Vishnoi, N. K., "Deterministic extractors for bit-fixing sources." Preliminary draft, December 2003.

[79] Lipton, R. J. and Vishnoi, N. K., "Deterministic identity testing for multivariate polynomials," in *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, no. 14, pp. 756–760, 2003.

[80] Lovász, L., "On determinants, matchings and random algorithms," in *Fundamentals of Computing Theory*, pp. 565–574, Akademia-Verlag, 1979.

[81] Lu, C.-J., Reingold, O., Vadhan, S., and Wigderson, A., "Extractors: Optimal up to constant factors," in *Proceedings of the ACM Symposium on the Theory of Computing*, no. 35, pp. 602–611, 2003.

[82] Lund, C., Fortnow, L., Karloff, H., and Nisan, N., "Algebraic methods for interactive proof systems," *Journal of the ACM*, vol. 39, pp. 859–868, 1992.

[83] Mansour, Y., "An $o(n^{\log\log n})$ learning algorithm for DNF under the uniform distribution," *Journal of Computer and System Sciences*, vol. 50, pp. 543–550, 1995.

[84] Micciancio, D., "The shortest vector in a lattice is hard to approximate to within some constant," *SIAM Journal of Computing*, vol. 30, no. 6, pp. 2008–2035, 2001.

[85] Micciancio, D., "Almost perfect lattices, the covering radius problem, and applications to Ajtai's connection factor," Tech. Rep. 66, ECCC, 2003.

[86] Micciancio, D. and Goldwasser, S., *Complexity of lattice problems: A cryptographic perspective.* Kluwer Academic Publishers, 2002.

[87] Minkowski, H., *Geometrie der zahlen.* Tuebner,: Leizpig, 1910.

[88] Mossel, E., O'Donnell, R., and Servedio, R., "Learning juntas," in *Proceedings of the ACM Symposium on the Theory of Computing*, no. 35, pp. 206–212, 2003.

[89] Motwani, R. and Raghavan, P., *Randomized Algorithms.* Cambridge University Press, 1995.

[90] Mulmuley, K., Vazirani, U. V., and Vazirani, V. V., "Matching is as easy as matrix inversion," *Combinatorica*, vol. 7, no. 1, pp. 105–113, 1987.

[91] Murty, M. R. and Esmonde, J., *Problems in Algebraic Number Theory.* Springer-Verlag, 1999.

[92] Naor, M. and Naor, S., "Small bias probability spaces: efficient constructions and applications," *SIAM Journal of Computing*, vol. 22, no. 4, pp. 838–856, 1993.

[93] Nisan, N. and Ta-Shma, A., "Extracting randomness: A survey and new constructions," *Journal of Computer and System Sciences*, vol. 58, no. 1, pp. 148–173, 1999.

[94] Nisan, N. and Wigderson, A., "Hardness vs. Randomness," *Journal of Computer and System Sciences*, vol. 49, pp. 149–167, 1994.

[95] Papadimitriou, C., *Computational Complexity*. Addison-Wesley, 1994.

[96] Pfister, A., "Zur darstellung definiter funktionen als summe von quadraten," *Invent. Math.*, vol. 4, pp. 229–237, 1967.

[97] Powers, V. and Reznick, B., "A new bound for Po'lya's Theorem with applications to polynomials positive on polyhedra," *J. Pure Appl. Alg.*, vol. 164, pp. 221–229, 2001.

[98] Prestel, A. and Delzell, C., *Positive Polynomials: From Hilbert's 17th Problem to Real Algebra*. Springer Monographs in Mathematics, Springer-Verlag, 2001.

[99] Raz, R., "A parallel repetition theorem," *SIAM Journal of Computing*, vol. 27, no. 3, pp. 763–803, 1998.

[100] Regev, O., "New lattice based cryptographic constructions," in *Proceedings of the ACM Symposium on the Theory of Computing*, no. 35, pp. 407–416, 2003.

[101] Reznick, B., "Some concrete aspects of Hilbert's 17th problem," *Publ. Math. Univ. Paris VII*, January 1996.

[102] Reznick, B., "On the absence of uniform denominators in Hilbert's seventeenth problem." Manuscript, 2004.

[103] Ribenboim, P., *The Book of Prime Number Records*. Springer-Verlag, 1989.

[104] Rivest, R., "All-or-nothing encryption and the package transform," in *Fast Software Encryption, 4th International Workshop, FSE '97*, vol. 1267 of *Lecture Notes in Computer Science*, pp. 210–218, Springer, 1997.

[105] Roy, M. F., "The role of Hilbert's problems in real algebraic geometry," in *Proceedings of the ninth EWM Meeting*, (Loccum, Germany), 1999.

[106] Saks, M., "Randomization and derandomization in space-bounded computation," in *Proceedings of the Annual IEEE Conference on Computational Complexity*, no. 11, pp. 128–149, 1996.

[107] Santha, M. and Vazirani, U. V., "Generating quasi-random sequences from semi-random sources," *Journal of Computer and System Sciences*, vol. 33, pp. 75–87, 1986.

[108] Schnorr, C., "A hierarchy of polynomial-time basis reduction algorithms," in *Proceedings of Conference on Algorithms*, (Péecs, Hungary), pp. 375–386, 1985.

[109] Schwartz, J., "Fast probabilistic algorithms for verification of polynomial identities," *Journal of the ACM*, vol. 27, pp. 701–717, 1980 1980.

[110] Shaltiel, R., *Recent developments in Extractors*, vol. 77, pp. 67–95. 2002.

[111] Shamir, A., "IP=PSPACE," *Journal of the ACM*, vol. 39, no. 4, pp. 869–877, 1992.

[112] Shparlinski, I., 2003. Private communication.

[113] Stengle, G., "A nullstellensatz and a positivstellensatz in semi-algebraic geometry," *Math. Ann.*, vol. 207, pp. 87–97, 1974.

[114] Thiele, R., "Hilbert's twenty-fourth problem," *American Math. Monthly*, vol. 110, no. 1, pp. 1–23, 2003.

[115] Trevisan, L. and Vadhan, S., "Extracting randomness from samplable distributions," in *Annual Symposium on Foundations of Computer Science*, no. 41, pp. 32–42, 2000.

[116] Valiant, L., "A theory of the learnable," *Communications of the ACM*, vol. 27, no. 11, pp. 1134–1142, 1984.

[117] Valiant, L. and Vazirani, V., "NP is as easy as detecting unique solutions," *Theoretical Computer Science*, vol. 47, no. 3, pp. 85–93, 1986.

[118] van Emde Boas, P., "Another NP-complete problem and the complexity of computing short vectors in a lattice," Tech. Rep. 81-04, Univ. of Amsterdam, Amsterdam, 1981.

[119] Verbeurgt, K., "Learning DNF under the uniform distribution in quasi-polynomial time," in *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pp. 314–326, Morgan Kaufmann, 1990.

[120] Verbeurgt, K., "Learning sub-classes of monotone DNF on the uniform distribution," in *Algorithmic Learning Theory, 9th International Conference* (Richter, M. M., Smith, C. H., Wiehagen, R., and Zeugmann, T., eds.), vol. 1501 of *Lecture Notes in Computer Science*, pp. 385–399, Springer, 1998.

[121] Vishnoi, N. K., "Self similarity of Pascal's Triangle with application to learning symmetric juntas." Manuscript, March 2004.

[122] von Neumann, J., "Various techniques used in connection with random digits," *App. Math. Ser.*, vol. 12, pp. 36–38, 1951.

[123] von zur Gathen, J. and Roche, J., "Polynomials with two values," *Combinatorica*, vol. 17, no. 3, pp. 345–362, 1997.

[124] Zippel, R., *Probabilistic Algorithms for Sparse Polynomials*. PhD thesis, MIT, 1979.